PERIODICA POLYTECHNICA
ELECTRICAL ENGINEERING

TECHNICAL UNIVERSITY OF BUDAPEST

Vol. 41. No. 3.
1997

PERIODICA POLYTECHNICA

A contribution to international technical sciences, published by the Technical University of Budapest in six separate series, covering the following sciences:

Chemical Engineering
Civil Engineering
Electrical Engineering and Informatics
Mechanical Engineering
Social and Management Sciences (Earlier: Humanities and Social Sciences)
Transportation Engineering

UNIVERSITY LEADERS:

Á. DETREKŐI, Rector Magnificus
J. GINSZTLER, Vice Rector for International Relations
GY. HORVAI, Vice Rector for Research Activities
F. VÖRÖS, Vice Rector for Education
B. PETRÓ, Dean of the Faculty of Architecture
M. KUBINYI, Dean of the Faculty of Chemical Engineering
GY. FARKAS Dean of the Faculty of Civil Engineering
L. PAP, Dean of the Faculty of Electrical Engineering and Informatics
K. MOLNÁR, Dean of the Faculty of Mechanical Engineering
GY. CSOM, Dean of the Faculty of Natural and Social Sciences
É. KÖVES-GILICZE, Dean of the Faculty of Transportation Engineering

GENERAL EDITOR OF PERIODICA POLYTECHNICA:
F. WETTL

Technical editor of Periodica Polytechnica:
M. Tarján-Fábry

ELECTRICAL ENGINEERING SERIES

Published quarterly

Contributions in electrical engineering and computer science should be sent to:
TECHNICAL UNIVERSITY OF BUDAPEST
Periodica Polytechnica
Electrical Engineering
H-1521 Budapest, HUNGARY
Telefax: + 36 1 463-3041; E-mail: perpol@tuo.bme.hu

For subscriptions please contact "Andreas Hess" Ltd. (Mail: P.O.B. 290, Budapest III, H–1300; Telefax: +36 1 250–2188) or its representatives abroad.

Exchange copies should be requested from the International Exchange Department of the Central Library of the Technical University of Budapest
(H–1111 Budapest, Budafoki út 4.; Telefax: + 36 1 463–2440).

Periodica Polytechnica Ser. Electrical Engineering is abstracted/indexed in: Science Abstracts (Electrical Engineering Abstracts, Computer and Control Abstracts), Engineering Index. Data bases: Inspec Electronics and Computing, Compendex Plus.

Published by the Technical University of Budapest, Hungary, with the financial help of the foundation "Ipar a korszerű mérnökképzésért" (Industry for Modern Education in Engineering).

HU ISSN: 0324-6000

PERIODICA POLYTECHNICA

ELECTRICAL ENGINEERING

Vol. 41 * No. 3 * 1997

TECHNICAL UNIVERSITY
BUDAPEST

PRINTED IN HUNGARY
LIGATURA LTD – ÁFÉSZ PRESS, VÁC

PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 41, NO. 3, PP. 167-183 (1997)

ON TWO-POINT RESOLUTION OF IMAGING SYSTEMS

A. J. DEN DEKKER

Department of Applied Physics
Delft University of Technology
P.O. Box 5046, 2600 GA Delft
The Netherlands
Tel: +31 0 15-278-1823
Fax: +31 0 15-278-4263
E-mail: dekker@duttncb.tn.tudelft.nl

Received: Nov. 28, 1995

Abstract

In this paper a new criterion is proposed for optical two-point resolution, applicable to coherent, incoherent, and partially coherent imaging. Unlike classical resolution criteria, such as Rayleigh's, the new criterion takes account of the presence of errors in the observed intensity distributions. Based on a parameter estimation approach, it shows how the resolvability of the imaged point sources depends on these errors. Additionally, a test for the resolvability of the point sources from a given set of observations is presented. Moreover, a procedure is proposed for the computation of the errors having minimum energy among all errors undermining the resolution. The results presented include, as a special case, earlier results on two-point resolution of strictly incoherent imaging systems.

Keywords: two-point resolution, imaging system, resolution limit, parameter estimation, partially coherent imaging.

1. Introduction

The two-point resolution of an imaging system, that is, its ability to resolve two point sources of equal intensity, is widely in use as a measure for the system's resolving capabilities. In the past, many resolution criteria have been proposed based on the assumption that the resolving power of an imaging system is limited by the diffraction at the aperture of the imaging lens of the system. Due to this diffraction, a point source is not imaged as a point image, but as the Fraunhofer diffraction pattern of the aperture. Hence, this diffraction pattern can be regarded as the point spread function of the imaging system. Of all the proposed criteria referred to above, the classical Rayleigh resolution criterion [1] is surely the most famous. According to the Rayleigh criterion, originally derived for incoherent imaging, two point sources are to be considered as just resolved if the central maximum of the intensity diffraction pattern produced by one point source coincides with the first zero of the intensity diffraction pattern produced by the other point source. This means that Rayleigh's resolution limit is

given by the distance between the central maximum and the first zero of the intensity point spread function of the imaging system concerned. In Rayleigh's considerations the human visual system has been employed in the role of a sensor used to detect differences in intensity at various points of the composite intensity distribution produced by the two point sources together. So, obviously, Rayleigh based his criterion not only on the properties of the diffraction-limited imaging system, but also on presumed resolving capabilities of the human visual system. Consequently, Rayleigh's resolution limit cannot be regarded as a law of physics, but rather as a rule of thumb. Other notable examples of resolution criteria, involving both the properties of the imaging system and the human visual system, are those of BUXTON, HOUSTON, SCHUSTER and SPARROW [2–3]. All these so called classical criteria have in common that they provide resolution limits that are completely set by the functional form of the point spread function of the diffraction-limited imaging system. No mention is made of the presence of errors in the observed intensity distributions.

Nowadays, it has been recognized for some time that diffraction does not impose an absolute limit to the resolving power of an imaging system. When visual inspection is replaced by intensity measurement, knowledge of the point spread function of the imaging system makes it possible to attain unlimited resolution, provided that the intensity measurements are noise free. For, on this condition, numerically fitting a mathematical model of point spread functions to the observed composite intensity distribution would make it possible to resolve the two point sources exactly.

When we turn to spatial frequency domain, viewing the imaging system as a linear spatial filter transforming the light field at the object plane into the light field at the image plane, the work on superresolution [4–6] has shown that in spite of the bandlimiting filter characteristics of the aperture of the diffraction-limited imaging system, knowledge of the point spread function, together with some reasonable a priori information about the object (e.g., knowledge that the object is of finite size which implies that its spatial Fourier transform is analytical) makes it possible to reconstruct the object exactly by applying mathematical operations on these spatial frequencies that do pass the aperture.

To sum up, it may be stated that, in the absence of measurement errors, an imaging system with a known point spread function can *theoretically* attain as high a resolving power as desired. *In practice*, however, the intensity measurements are always disturbed by noise (non-systematic errors). Furthermore, the point spread function will rarely be known exactly, which means the introduction of systematic errors. It may be obvious that it will be these errors, both systematic and non-systematic, that prevent an infinite degree of resolution. This insight necessitates a reevaluation of

two-point resolution in order to establish a resolution criterion that, unlike classical resolution criteria, takes into account the presence of errors in the observed intensity distribution. Such a resolution criterion is proposed in this paper.

The theory developed in this paper is related to earlier work by VAN DEN BOS on resolution in model-fitting [8–10]. In references [8] and [9] the two-point resolution of fitting a nonlinearly parametric sum model of intensity point spread functions to intensity observations was considered for one-dimensional and two-dimensional imaging systems respectively. These results are only applicable to incoherent imaging. The present study generalizes the results and extends them to include models used to describe one-dimensional partially and fully coherent imaging, which are *no* sum models.

The main results may be summarized as follows. A mathematical model of amplitude point spread functions, which is assumed to underlie the intensity observations, is fitted in least-squares sense to these error corrupted observations with respect to the intensities and the locations of the two point sources. The locations are non-linearly present in the model to be fitted. Now, the two point sources have been defined as being resolved if and only if the model-fitting solutions for their locations are distinct. It has been shown that the Euclidean space of the errors, or equivalently that of the observations, can be divided into two regions, separated from each other by a hypersurface called the bifurcation set. In the one region two distinct solutions for the locations are found, while in the other region the solutions exactly coincide. Thus defined, the bifurcation set represents the error limit to resolution achievable by model-fitting. Furthermore, it has been found to be relatively easy to decide on which side of the bifurcation set a given set of observations is located. This offers the experimenter the possibility to determine beforehand whether or not the two point sources can be resolved from the available observations. Additionally, an operational procedure has been developed to compute the errors that have minimum energy (in the sense of their Euclidean norm) among all errors causing coincidence of the solutions. This minimum energy offers the experimenter a scalar criterion to determine to what extent coincidence of the solutions is to be expected for assumed error energy. Besides, it can be used to compare the resolving capabilities of different imaging systems.

This paper is organized as follows. In section 2 the model to be fitted to the intensity observations is described. As a model-fitting criterion the least-squares criterion is chosen, since it is most frequently used in practice. In section 3 the possible structures of the model-fitting criterion under influence of the errors are discussed. The structure of the criterion is decisive for the kind of solution. In section 4 the resolution discriminant and the bifurcation set are derived. Section 5 introduces the concept of

critical errors. In section 6 an illustrative numerical example is presented. In section 7 conclusions are drawn.

2. Fitting a Model to the Observed Intensity Distribution

Consider an object consisting of two point sources of light. Let this two-point object be imaged by a one-dimensional diffraction-limited imaging system. Assume that the two point sources A and B have equal intensity and that they are located at the positions $x = \beta_1$ and $x = \beta_2$ with respect to the optical axis, where x is the coordinate in the object plane. Due to the diffraction at the aperture of the imaging lens, each point source is not imaged as an image point, as predicted by geometrical optics, but as the Fraunhofer diffraction pattern of the aperture. Let $h(x')$ be the Fraunhofer amplitude diffraction pattern, i.e., *the amplitude point spread function (apsf)* of the imaging system concerned, where x' is the coordinate in the image plane. Then, the composite intensity distribution in the image plane produced by the two point sources together is given by:

$$g(x';\alpha,\beta_1',\beta_2') = \alpha[|h(x'-\beta_1')|^2 + |h(x'-\beta_2')|^2 + 2\Re\{\gamma_{12}\cdot h(x'-\beta_1')\cdot h^*(x'-\beta_2')\}] , \tag{1}$$

where β_1' and β_2' are the locations of the geometrical image points of A and B respectively. $\Re$ is the real operator, * denotes the complex conjugate, γ_{12} is the complex degree of coherence [11] and α is the amplitude, which is proportional to the intensity of the point sources. It may be shown [11] that $|\gamma_{12}| \leq 1$. From now on it will be assumed that the *apsf* is real. Let $\gamma = \Re\{\gamma_{12}\}$. The value $\gamma = 0$ corresponds to mutually fully incoherent point sources, $\gamma = +1$, -1 imply completely cophasal and antiphasal coherent point sources respectively, whereas $|\gamma| < 1$ implies partially coherent point sources.

Suppose that N observations $w_1,\ldots,w_N$ have been made on $g(x;\alpha,\beta_1',\beta_2')$, defined by:

$$w_n = g_n(\alpha,\beta_1',\beta_2') + v_n , \quad n = 1,\ldots,N \tag{2}$$

with $g_n = g(x_n';\alpha,\beta_1',\beta_2')$, where the x_n' are exactly known values of x' (the measurement points). The v_n are the errors in the observations. Systematic errors are defined as the expectation $E[v_n]$ of the v_n, whereas non-systematic errors are defined as $v_n - E[v_n]$. The errors are assumed to be

small in comparison with the errorless observations. Now, in this paper, two-point resolution will be defined as the ability of the imaging system, including detection and digital computing facilities, to obtain two distinct solutions for the location parameters b_1 and b_2 of the model $g(x'_n; a, b_1, b_2)$ when the latter is fitted to the observations, with respect to $\mathbf{t} = (a, b_1, b_2)^T$. If a least-squares procedure is used to estimate the parameters, the model-fitting solutions for the amplitude and the locations are equal to the solution of the following minimization problem:

$$\underset{a,b_1,b_2}{\text{minimize}}\, J_2(a, b_1, b_2) = \sum_n d_n^2(a, b_1, b_2) \ , \tag{3}$$

where $J_2(a, b_1, b_2)$ is the least-squares criterion and the deviations $d_n(a, b_1, b_2)$ are defined by:

$$d_n(a, b_1, b_2) = w_n - g_n(a, b_1, b_2) \ . \tag{4}$$

The least-squares solution is the absolute minimum of $J_2(a, b_1, b_2)$. A necessary condition for a minimum is that it must be a stationary point of the criterion. Stationary points are defined as points where the gradient of the criterion is equal to zero. So, from *Eqs.* (1–4) it follows that the stationary points $(\hat{a}, \hat{b}_1, \hat{b}_2)$ of $J_2(a, b_1, b_2)$ satisfy the so-called normal equations given by:

$$\sum_n d_n(\hat{a}, \hat{b}_1, \hat{b}_2)[h_n^2(\hat{b}_1) + h_n^2(\hat{b}_2) + 2\gamma h_n(\hat{b}_1)h_n(\hat{b}_2)] = 0 \ , \tag{5}$$

$$\sum_n d_n(\hat{a}, \hat{b}_1, \hat{b}_2)[h_n(\hat{b}_1)h_n^{(1)}(\hat{b}) + \gamma h_n^{(1)}(\hat{b}_1)h_n(\hat{b}_2)] = 0 \ , \tag{6}$$

$$\sum_n d_n(\hat{a}, \hat{b}_2, \hat{b}_2)[h_n(\hat{b}_2)h_n^{(1)}(\hat{b}_2) + \gamma h_n(\hat{b}_1)h_n^{(1)}(\hat{b}_2)] = 0 \ , \tag{7}$$

where $h_n(\hat{b}_k) = h(x'_n - \hat{b}_k)$ and $h^{(\ell)}(\hat{b}_k)$ is the ℓ-th order derivative of $h_n(b_k)$ with respect to b_k evaluated at $\hat{b}_k$.

Next, suppose that a model describing the intensity distribution as if it was produced by *one* point source is fitted to the same observations. The least-squares criterion for this so-called *first order* model $a^* h^2(b)$ is given by:

$$J_1(a^*, b) = \sum_n d_{n,1}^2(a^*, b) \ , \tag{8}$$

with

$$d_{n,1}(a^*, b) = w_n - a^* h_n^2(b) \ . \tag{9}$$

The stationary points $(\hat{a}^*, \hat{b})$ of $J_1(a^*, b)$ with respect to a^* and b satisfy

$$\sum_n d_{n,1}(\hat{a}^*, \hat{b}) h_n^2(\hat{b}) = 0 \ , \tag{10}$$

$$\sum_n d_{n,1}(\hat{a}^*, \hat{b}) h_n(\hat{b}) h_n^{(1)}(\hat{b}) = 0 \ . \tag{11}$$

From the *Eqs.* (5–7) and (10–11), it can be concluded that all points $(\hat{a}, \hat{b}_1, \hat{b}_2)$, with $\hat{b}_1 = \hat{b}_2 = \hat{b}$ and $\hat{a} = \hat{a}^*/(2 + 2\gamma)$ are stationary points of $J_2(a, b_1, b_2)$. In this way, a stationary point of J_1 generates a stationary point of J_2. So, the stationary points of J_2 can be divided into a group that contains the stationary points for which the location parameters b_1 and b_2 are distinct and a group that contains the stationary points characterized by *exactly* coinciding location parameters. Stationary points belonging to the first group and the last group will be called *second order* and *first order stationary points* respectively.

It may be obvious that the structure of the criterion J_2 is decisive for the kind of solution and by that, ultimately, for the resolvability of the model-fitting solutions for the location parameters. In the next section we will analyze this structure and the influence of the errors upon it.

3. The Structure of the Least-Squares Criterion

3.1. Stationary Points of the Criterion

For the purpose of this paper the location parameters β_1' and β_2' are considered to be very close, so that difficulties with resolution may be expected. For the moment, let's consider errorless observations. Then, for reasons of symmetry, it will be clear that the least-squares criterion J_2 has two closely spaced absolute minima $(\alpha, \beta_1', \beta_2')$ and $(\alpha, \beta_2', \beta_1')$, at which the criterion is equal to zero. Since the location parameters are close to each other, a first order model will also fit quite well to the observations in the sense of least-squares. So, intuitively, least-squares fitting of the first order model will result in a so-called *first order solution* $(\hat{a}^*, \hat{b})$, , where $\hat{a}^*$ and $\hat{b}$ will approximately be equal to $(2 + 2\gamma)\alpha$ and the average of β_1' and β_2' respectively. Now, as we saw in the preceding section, this first order solution $(\hat{a}^*, \hat{b})$ generates a stationary point $(\hat{a}, \hat{b}, \hat{b})$, with $\hat{a} = \hat{a}^*/(2 + 2\gamma)$, of J_2. So, in between the two (second order) minima, the criterion J_2 will have an extra (first order) stationary point. This first order stationary point is a one-saddle point. This can be seen as follows: since the criterion $J_1(a^*, b)$ is the intersection of the plane $b_1 = b_2$ and the (appropriately scaled) criterion $J_2(a, b_1, b_2)$ and $(\hat{a}^*, \hat{b})$ is the minimum of $J_1(a^*, b)$, the first order

stationary point must be a minimum in all directions but $b_1 - b_2$. Travelling on the criterion in parameter space in the latter direction from the absolute minimum $(\alpha, \beta_1', \beta_2')$ through the first order stationary point $(\hat{a}, \hat{b}, \hat{b})$ to the absolute minimum $(\alpha, \beta_2', \beta_1')$ on the other side of the plane $b_1 = b_2$, means going uphill from $(\alpha, \beta_1', \beta_2')$ to $(\hat{a}, \hat{b}, \hat{b})$ and downhill from there. In conclusion, the first order stationary point is a maximum in the direction $b_1 - b_2$, but a minimum in all other directions. The minima are close to the saddle point, since the location parameters were assumed close.

A remarkable phenomenon appearing from simulation experiments is the fact that errors in the observations may change the structure of the criterion in such a way that the two minima and the saddle point merge into one minimum at the first order stationary point $(\hat{a}, \hat{b}, \hat{b})$. Then resolution of the two point sources is no longer possible, since the least-squares solutions for the location parameters exactly coincide. In the next subsection this coincidence phenomenon will be studied by investigating the influence of the errors on both the nature of the first order stationary point and the behavior of the criterion around it.

3.2. Taylor Expansion of the Criterion

The least-squares criterion J_2 is Taylor expanded around the first order stationary point $\hat{\mathbf{t}} = (\hat{a}, \hat{b}, \hat{b})^T$. The constant term is equal to $J_2(\hat{a}, \hat{b}, \hat{b})$, or equivalently $J_1(\hat{a}^*, \hat{b})$. The linear terms are absent, since the origin of the Taylor expansion is a stationary point. The quadratic terms are given by:

$$\frac{1}{2}\, {}^1\hat{\mathbf{t}}\mathbf{H}_2\, {}^1\hat{\mathbf{t}}^T\,, \tag{12}$$

where ${}^1\mathbf{t} = \mathbf{t} - \hat{\mathbf{t}}$. $\mathbf{H}_2$ is the 3×3 Hessian matrix of J_2 with respect to $\mathbf{t} = (a, b_1, b_2)^T$ at $\hat{\mathbf{t}} = (\hat{a}, \hat{b}, \hat{b})^T$. The elements of H_2 are defined as:

$$(H_2)_{ij} = \frac{\partial^2 J_2(\hat{\mathbf{t}})}{\partial t_i \partial t_j}\,, \quad i, j = 1, \ldots, 3\,. \tag{13}$$

The eigenvalues of the Hessian matrix determine the nature of the stationary point. In order to get more insight in the eigenstructure of $\mathbf{H}_2$, the parameters are subsequently linearly transformed into

$${}^2a = {}^1a(2 + 2\gamma)\,, \quad {}^2b_1 = \frac{1}{2}({}^1b_1 + {}^1b_2)\,, \quad {}^2b_2 = \frac{1}{2}({}^1b_1 - {}^1b_2)\,. \tag{14}$$

It may be shown that in these coordinates the Taylor expansion becomes

$$J_2({}^2\mathbf{t}) = J_2(\hat{\mathbf{t}}) + \frac{1}{2}\,{}^2\mathbf{t}^T \mathrm{diag}\,(\mathbf{H}_1\ \rho_2)^2\mathbf{t} + O({}^2\mathbf{t}^3)\,, \tag{15}$$

where $\mathbf{H}_1$ is the 2×2 Hessian matrix of $J_1(a^*, b)$ at the first order solution $(\hat{a}^*, \hat{b})$. The symbol $O(\mathbf{t}^n)$ is the order symbol of Landau. It represents all terms of degree n and higher. The element ρ_2 is given by:

$$\rho_2 = -4\hat{a}^* \left[\frac{(1-\gamma)}{(1+\gamma)} \sum_n d_{n,1}(\hat{a}^*, \hat{b})(h_n^{(1)}(\hat{b}))^2 + \sum_n d_{n,1}(\hat{a}^*, \hat{b}) h_n(\hat{b}) h_n^{(2)}(\hat{b}) \right] . \tag{16}$$

By the assumption that the first order solution $(\hat{a}^*, \hat{b})$ always exists, i.e., that the first order model fits well to the observations, independent of the particular realization of the errors, the Hessian matrix $\mathbf{H}_1$ is always positive definite. Then all eigenvalues of $\mathbf{H}_1$ are positive. Hence, the nature of the stationary point $\hat{\mathbf{t}}$ is completely set by the sign of ρ_2, since this sign corresponds to that of the remaining eigenvalue of $\mathbf{H}_2$. The point $\hat{\mathbf{t}}$ is a one-saddle, a non-degenerate minimum or a degenerate minimum as ρ_2 is negative, positive or zero respectively.

The analysis of the higher order terms of the Taylor expansion can be simplified drastically by using the catastrophe theory. Catastrophe theory [12] is concerned with the structural change of a parametric function under influence of its parameters. It tells us that a structural change of the function is always preceded by a degeneracy of one of its stationary points. The theory also shows that the independent variables of the function can be split into *essential* and *inessential* variables. The essential variables correspond to the directions in which degeneracy occurs. The number of essential variables is equal to the number of possible vanishing eigenvalues of the Hessian matrix of the function at the stationary point that may become degenerate. In order to analyze structural change, the parametric function can be replaced by a Taylor expansion in the essential variables, around the latter stationary point. Terms in the inessential variables do not play a role at all in the structural change. According to catastrophe theory the global structure of a parametric function with only one essential variable is completely set by its Taylor expansion up to the degree of which coefficient cannot vanish under the influence of its parameters. In most practical applications, including the one considered in this paper, the required degree of the Taylor polynomial is very low.

The parametric function studied in this paper is the least-squares criterion as a function of the model parameters. Its parameters are the errors in the observations. *Eq.* (15) shows that only the eigenvalue corresponding to the coordinate ${}^2b_2$ may vanish. So, ${}^2b_2$ is the only essential variable. Since $\mathbf{H}_1$ is symmetric, it can be diagonalized by a not specified non-singular linear transformation of the coordinates ${}^2a$ and ${}^2b_1$ into ${}^3a$ and

${}^3b_1$. Although this transformation does not effect the essential coordinate ${}^2b_2$, the superscript ^{3} is also used for ${}^3b_2 = {}^2b_2$. Now, the quadratic terms of the Taylor expansion can be described as:

$$\frac{1}{2}\lambda_1' \, {}^3a^2 + \frac{1}{2}\lambda_2' \, {}^3b_1^2 + \frac{1}{2}\rho_2 \, {}^3b_2^2 \, , \tag{17}$$

where λ_1' and λ_2' are the eigenvalues of $\mathbf{H}_1$. Next, the cubic and quartic terms of the Taylor expansion are considered. The coefficient of ${}^3b_2^3$ happens to be equal to zero, because of the symmetry of the model. The cubic cross terms containing the inessential coordinates $({}^3a, {}^3b_1)$ are removed by applying a procedure described by [12]. First, all cubic terms in which ${}^3a$ appears are collected. Let the sum of these terms be $\lambda_1' \, {}^3aQ_1 \updownarrow Q_1$ is then homogeneously quadratic. The sum of $\frac{1}{2}\lambda_1' \, {}^3a^2$ and these terms may be written as:

$$\frac{1}{2}\lambda_1'({}^3a + Q_1)^2 - \frac{1}{2}\lambda_1' Q_1^2 \, , \tag{18}$$

where the last term is homogeneously quartic. This procedure is next applied to all remaining cubic terms containing ${}^3b_1$, which can be described as $\lambda_2' \, {}^3b_1 Q_2$. Subsequently, the coordinates ${}^3a$ and ${}^3b_1$ are nonlinearly transformed into the following curvilinear coordinates:

$${}^4a = {}^3a + Q_1 \, , \quad {}^4b_1 = {}^3b_1 + Q_2. \tag{19}$$

The coordinate ${}^3b_2$ is again not affected by this transformation: ${}^4b_2 = {}^3b_2$. Substituting *Eq.* (19) in the Taylor expansion removes all cubic terms. Notice that the quartic terms resulting from this procedure are described by:

$$-\frac{1}{2}\sum_m \lambda_m' Q_m^2 \, , \quad m = 1, 2. \tag{20}$$

Also notice that the quartic terms (20) and those already present in the original expansion are still in the old coordinates. To change this, *Eq.* (19) is used to express the old coordinates as a power series in the new ones and the result is substituted in the quartic and higher order terms. Now, the coefficient τ of ${}^4b_2^4$ described by:

$$\tau = \nu - \frac{1}{2}(\eta_1^2/\lambda_1') - \frac{1}{2}(\eta_2^2/\lambda_2') \, , \tag{21}$$

where ν, η_1 and η_2 are the coefficients of ${}^3b_2^4$, ${}^3a \, {}^3b_2^2$ and ${}^3b_1 \, {}^3b_2^2$ respectively in the original expansion. Without derivation, the expression for ν is given

by:

$$\nu = \hat{a}^{*2}\left[\frac{(1-\gamma)^2}{(1+\gamma)^2}H_{040} + H_{202} + 2\frac{(1-\gamma)}{(1+\gamma)}H_{121}\right] - \frac{1}{6}\hat{a}^*\left[3D_{00200} + 4\frac{(1-\gamma)}{(1+\gamma)}D_{01010} + D_{10001}\right] , \tag{22}$$

where

$$H_{klm} = \sum_n h_n^k (h_n^{(1)})^l (h_n^{(2)})^m \tag{23}$$

and

$$D_{ijklm} = \sum_n d_n h_n^i (h_n^{(1)})^j (h_n^{(2)})^k (h_n^{(3)})^l (h_n^{(4)})^m , \tag{24}$$

with $h_n = h_n(\hat{b})$, $h_n^{(l)} = h_n^{(l)}(\hat{b})$ and $d_n = d_{n,1}(\hat{a}^*, \hat{b})$. Also without derivation, the expressions for η_1 and η_2 are given by:

$$\eta_1 = 2\hat{a}^*\left[H_{301} + \frac{(1-\gamma)}{(1+\gamma)}H_{220}\right] + 2\left[\frac{1}{(1+\gamma)}D_{02000} - D_{10100}\right] , \tag{25}$$

and

$$\eta_2 = \Phi - \left(2\hat{a}^*\frac{H_{310}}{H_{400}}\right)\eta_1 , \tag{26}$$

where

$$\Phi = 4\hat{a}^{*2}\left[H_{211} + \frac{(1-\gamma)}{(1+\gamma)}H_{130}\right] - 2\hat{a}^*\left[\frac{1}{2}D_{00200} + D_{10010} + \frac{(3-\gamma)}{(1+\gamma)}D_{01100}\right] . \tag{27}$$

Next, a similar procedure could be used to remove all quartic terms in which ${}^4a$ and ${}^4b_1$ appear and so on. Notice that such a procedure would not further affect the coefficient τ. So, the resulting Taylor expansion would be of the form

$$\lambda_1(a')^2 + \lambda_2(b_1')^2 + \rho(b_2')^2 + \tau(b_2')^4 + O(\mathbf{t}'^5) , \tag{28}$$

where $\lambda_1 = \frac{1}{2}\lambda_1'$, $\lambda_2 = \frac{1}{2}\lambda_2'$ and $\rho = \frac{1}{2}\rho_2$. The new coordinates a' and b_1' have been obtained by applying the procedure described above to the quartic terms and $b_2' = {}^4b_2 = b_1 - b_2$.

In order to investigate whether or not the Taylor expansion up to the fourth degree in the essential variable is sufficient, let's consider the possible signs of the coefficients ρ and τ. As we saw before, for errorless observations the first order stationary point will be a saddle point, so ρ will be negative. However, considering *Eq.* (16), it can easily be understood that already relatively small errors in the observations may make ρ positive and thus the stationary point a minimum. This can be seen as follows. The *apsf* h and its first and second order derivatives are relatively independent of the errors, since the first order solution $(\hat{a}^*, \hat{b})$ will hardly change under influence of the errors. The derivations d_n on the other hand, depend strongly on the particular realization of the errors; relatively small errors are required to change their sign and by that the sign of ρ. From *Eqs.* (21-27), it follows that τ consists of a sum of terms that do not contain the deviations d_n and a sum of terms that do contain these deviations. The first sum of terms only depends on the first order solution $(\hat{a}^*, \hat{b})$ and is therefore relatively independent of the particular realization of the errors. It may be shown that this sum of terms is positive. The second sum consists of terms of order $|\mathbf{d}|_1$ and $|\mathbf{d}|_2$, where $|\mathbf{d}|_p$ denotes the ℓ_p norm of the vector $\mathbf{d} = (d_1, d_2, \ldots d_N)^T$. These terms will be negligible compared to the first sum of terms, since, by assumption, the first order model fits well to the observations. So, τ consists of a nearly constant positive term and terms of order $|\mathbf{d}|_1$ and $|\mathbf{d}|_2$ and will therefore always be positive. Consequently, the expansion (28) is sufficient to describe the possible structures of the least-squares criterion. This means that the study of the least-squares criterion under influence of the errors can be replaced by a study of the following quartic Taylor polynomial in the essential variable b_2':

$$\rho(b_2')^2 + \tau(b_2')^4 \ , \tag{29}$$

where the constant term has been omitted, since it does not influence the structure.

4. Resolution Discriminant and Bifurcation Set

The structure of the quartic Taylor polynomial (29) is fully established by the number and the nature of its stationary points. From *Eq.* (29) it follows, that the polynomial is always stationary at the point $b_2' = b_1 - b_2 = 0$. This stationary point is a maximum, a non-degenerate minimum or a degenerate minimum, if ρ is negative, positive or equal to zero respectively. These correspond to a one-saddle point, a non-degenerate minimum and a degenerate minimum of the least-squares criterion J_2 at the origin of expansion (29), i.e., at the first order stationary point $(\hat{a}, \hat{b}, \hat{b})$. Further analysis of *Eq.* (29) shows that the criterion will have two additional stationary

points if $\frac{\rho}{\tau}$ is negative. Given the positiveness of τ, it may be shown that these two additional stationary points are minima. To sum up, it can be concluded that if ρ is negative the criterion has locally two (second order) minima and one (first order) saddle point. This structure corresponds to the one for errorless observations, found on intuitive grounds in Section 3. In this case one will find two distinct least-squares solutions for the location parameters, so that resolution is guaranteed. If, however, under influence of the errors in the observations, ρ becomes positive, the first order stationary point changes from a saddle point into a minimum, while the two original minima vanish through coincidence with this first order stationary point. In this case the first order stationary point, at which $b_1 = b_2$, has become the absolute minimum of the criterion. Then the least-squares solutions for the location parameters will exactly coincide so that resolution is no longer possible. Simulation experiments have confirmed the existence of the two structures described above and up till now no local structures have been found different from these. Since the sign of ρ is decisive for the resolvability of the two point sources, ρ is called the *resolution discriminant.* To test a given set of observations with respect to resolvability, only ρ has to be computed. From *Eq.* (16) it follows that, for this purpose, fitting of a first order model suffices. Notice, that the resolution test is applicable to any value of γ, except for $\gamma = -1$.

In the Euclidean N space of the errors the subset of all points for which $\rho = 0$ separates both structures described above. The errors belonging to this subset and the corresponding first order solution $(\hat{a}^*, \hat{b})$ must satisfy the normal equations for $J_1(a^*, b)$ with respect to (a^*, b), and, in addition, the equation $\rho = 0$. This is a set of three equations in $N + 2$ variables, or, equivalently one equation in the N errors. This equation constitutes a subspace of the Euclidean N space of errors of codimension 1, which is called a *bifurcation set* in catastrophe theory. It divides the error space into a region, including the origin, where the point sources can be resolved, and a region where resolution is impossible. Thus defined, the bifurcation set represents the error limit to resolution achievable by model-fitting. Since the first order solution will hardly change under influence of the errors and the condition $\rho = 0$ is linear in the errors, the bifurcation set is approximately a hyperplane. Simulation experiments have affirmed the high accuracy of this approximation. Notice that the error space can easily be transformed into that of the observations by a simple translation.

When the functional form of the *apsf* is not identical to the one underlying the observations, modelling errors are introduced. These systematic errors may, possibly in combination with the additive errors v_n, also cause coincidence of the model-fitting solutions. This means that exact observations, i.e. all $v_n = 0$, do not guarantee resolution. It may be shown

that fitting a wrong model causes a shift of the bifurcation set in the error space. Nevertheless, the resolution test derived in this section applies to fitting a wrong model, too.

5. Critical Errors

Among all errors belonging to the bifurcation set, those that have minimum energy, in the sense of their Euclidean norm, are called the *critical errors*. Errors having smaller energy than the critical energy cannot cause coincidence of the solutions, errors having higher energy may do so. Knowledge of the critical energy offers the experimenter the possibility to find out to what extent resolution is to be expected for assumed error energy. Furthermore, it can be used to test beforehand whether or not the experimental set-up is good enough to meet the demands. Additionally, it can be used to compare the resolving capabilities of different imaging systems used for the same purpose. By computing the critical energy as a function of the distance between the two closely located point sources, a relation can be established between this distance and the maximum allowable energy of the errors.

The critical errors can exactly be found by minimizing the quadratic sum of all errors with respect to $(v_1, \ldots, v_N, a^*, b)$ subject to equality constraints (10), (11) and $\rho = 0$. This nonlinear minimization problem under nonlinear equality constraints can be solved by nonlinear programming methods and software. Since, however, this is an enormous, time-consuming task, an alternative procedure will be proposed in this section, providing an approximate expression for the critical errors that has proved to be very accurate. Compared to numerically finding a solution, this approximation means a drastic simplification.

Notice, that the minimization problem in consideration is a Lagrange problem whose solution must be under the stationary points $(\hat{v}_1, \ldots, \hat{v}_N, \hat{\mu}, \hat{\kappa}, \hat{\xi}, \hat{a}^*, \hat{b})$ of the Lagrange function

$$L = \sum_N v_n^2 + \mu \sum_n d_n h_n^2 + \kappa \sum_n d_n h_n h_n^{(1)} + \xi \left[\frac{(1-\gamma)}{(1+\gamma)} \sum_n d_n (h_n^{1)})^2 + \sum_n d_n h_n h_n^{(2)} \right], \tag{30}$$

where μ, κ and ξ are the Lagrange multipliers. Here and in what follows $h_n, h_n^{(\ell)}$ and d_n are defined as in *Eqs.* (22-24). Differentiating L with respect

to the v_n and equating the result to zero yields

$$\hat{v}_n = -\frac{1}{2}\mu h_n^2 - \frac{1}{2}\kappa h_n h_n^{(1)} - \frac{1}{2}\xi\left[\frac{(1-\gamma)}{(1+\gamma)}(h_n^{1)})^2 + h_n h_n^{(2)}\right] . \tag{31}$$

Subsequent differentiation of L with respect to the Lagrange multipliers and equating the result to zero yields, of course, the *Eqs.* (10), (11) and $\rho = 0$. Substituting (31) for the errors in these equations, produces equations identical to the normal equations for minimizing the deviations

$$\hat{d}_n = g_n(\alpha, \beta_1', \beta_2') + \hat{v}_n - \hat{a}^* h_n^2(\hat{b}) \tag{32}$$

in least-squares sense with respect to the Lagrange multipliers, which are, by *Eq.* (31), linearly present in the v_n. If $\{g_n(\alpha, \beta_1', \beta_2') - a^* h^2(b)\}$ is Taylor expanded about $(\hat{a}^*, \hat{b})$ in powers of $((2+2\gamma)\alpha - \hat{a}^*)$, $(\beta_1' - \hat{b})$ and $(\beta_2' - \hat{b})$, it can easily be shown that this least-squares minimization makes the deviations quadratic in the elements of $((2+2\gamma)\alpha - \hat{a}^*)$, $(\beta_1' - \hat{b})$ and $(\beta_2' - \hat{b})$. So, it can be concluded that for critical errors $\hat{d}_n \approx 0$. Hence, it follows from *Eq.* (32) that the critical errors are approximately equal to the additive inverse of the deviations of the exact observations and the first order model best fitting to the observations corrupted by the critical errors. Finally, differentiating L with respect to (a^*, b) and equating the result to zero shows that $(\hat{a}^*, \hat{b})$ must be approximately, but very accurately, equal to the solution that would have been obtained in the absence of errors. This means that the critical errors are approximately equal to the additive inverse of the deviations of the best fitting first order model to exact observations. The proof will be left out here, since it requires more or less the same consecutive steps as described in [10]. In this reference an approximate solution is derived for the critical errors associated with coincidence of model-fitting solutions for closely located nonlinear parameters of weighted-sum models, which is a topic closely related to the one described in this section.

Since the bifurcation set is approximately a hyperplane, it can be shown that an arbitrary error signal $\mathbf{v} = (v_1, \ldots, v_N)^T$ is not on the same side of the bifurcation set as the origin if [13]

$$\mathbf{v} \cdot \mathbf{e} \geq \mathbf{e} \cdot \mathbf{e} , \tag{33}$$

where $\mathbf{v} \cdot \mathbf{e}$ is the inner product of $\mathbf{v}$ and the critical error vector $\mathbf{e} = (e_1, \ldots, e_N)^T$. The inner product $\mathbf{e} \cdot \mathbf{e}$ is equal to the critical energy. Rule (33) can be used to find out whether or not for arbitrary, given errors coincidence of the solutions for the locations occurs, or for statistical errors, how probable this coincidence is [13].

6. Numerical Example

The purpose of this numerical example is to give an impression of the magnitude of the critical energy as a function of γ. For this purpose the critical energy is computed for the model

$$\alpha \left[h^2(x'_n - \frac{1}{2}\Delta\beta') + h^2(x'_n + \frac{1}{2}\Delta\beta') + 2\gamma h(x'_n - \frac{1}{2}\Delta\beta')h(x'_n + \frac{1}{2}\Delta\beta') \right] , \tag{34}$$

where $\alpha = 1$ and $h(x')$ is a gaussian shaped *apsf*: $h(x') = \exp\{-(1/2)x'^2\}$. This is done for $-1 < \gamma \leq 1$. The measurement points are described by $x'_n = -0.01 + (n - 11) \times 0.4$, $n = 1, \ldots, 21$. *Fig. 1* shows the square root of the critical energy as a fraction of $\hat{a}^*$ as a function of γ, respectively for $\Delta\beta' = 0.02$, $\Delta\beta' = 0.04$ and $\Delta\beta' = 0.08$. The results show that coincidence may already occur for relatively low error energy. They also show that resolution decreases with decreasing $\Delta\beta'$, as might be expected. Moreover, *Fig. 1* demonstrates that resolution decreases with increasing γ; undermining the resolution requires for fully cophasal coherent point sources errors that are, roughly, three times smaller than for fully incoherent point sources.

7. Discussion and Conclusions

An earlier presented theory with respect to the concept of optical two-point resolution, which was applicable to strictly incoherent imaging, has been extended to include coherent and partially coherent imaging as well. A new resolution criterion has been proposed that, unlike classical resolution criteria, takes account of the errors in the observed intensity distributions and what's more, it's even based on these errors. The criterion states that which errors make resolution impossible and which errors do not, depends on the distance between the two point sources, the intensity of the point sources, the choice of measurement points, the degree of coherence and the point spread function of the imaging system. A resolution test has been derived that offers the experimenter the possibility to determine beforehand whether or not the two point sources can be resolved from error-corrupted observations. Also a procedure has been presented to compute the errors having minimum energy among all errors undermining resolution. This minimum energy provides the experimenter with a scalar criterion to determine to what extent resolution is to be expected for the point spread

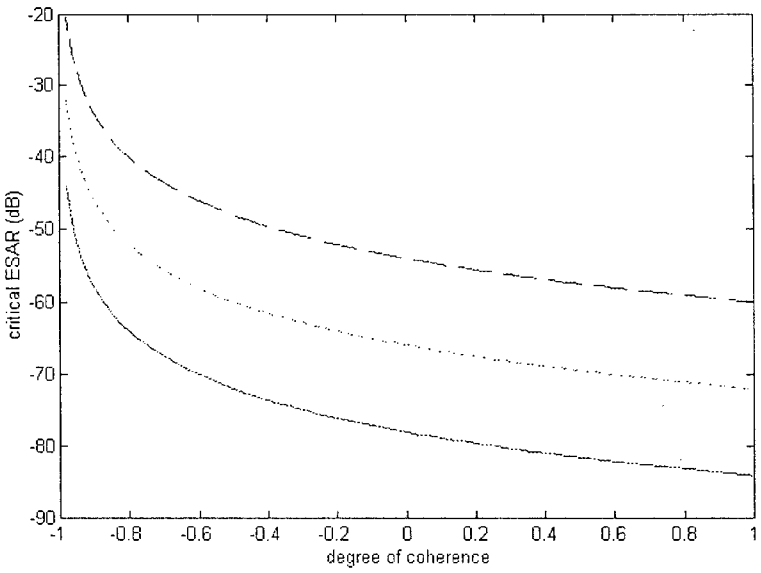


Fig. 1. Critical Error-to-Signal Amplitude Ratio (ESAR) for model (34) as a function of γ, for $\Delta\beta' = 0.02$ (solid), $\Delta\beta' = 0.04$ (dotted) and $\Delta\beta' = 0.08$ (dashed)

function assumed to underlie the observations and, perhaps a guess of, the error energy. Additionally, it can be used to compare the resolving capabilities of different instrumental set-ups used for the same purpose.

Finally, the author has found that the theory presented here is not confined to strictly one-dimensional imaging systems. For two-dimensional imaging systems a so called *resolution matrix* can be derived of which the smallest eigenvalue plays the role of resolution discriminant. These results will be reported later.

References

1. Lord RAYLEIGH, title of: STRUTT, J. W. (1899): *Scientific Papers*, Vol. 1, p. 415, Cambridge University Press, Cambridge,UK.
2. RAMSAY, B. P. – CLEVELAND, E. L. – KOPPIUS, O. T. (1941): *Journal of the Optical Society of America*, Vol. 31 (8), p. 26.
3. SPARROW, G. (1916): On Spectroscopic Resolving Power, *Astrophys. J.*, Vol. 44, p. 76.
4. HARRIS, J. L. (1964): Diffraction and Resolving Power, *Journal of the Optical Society of America*, Vol. 54 (7), pp. 931–936.
5. RUSHFORTH, C. K. – HARRIS, R. W. (1968): Restoration, Resolution and Noise, *Journal of the Optical Society of America*, Vol. 58 (4), pp. 539–545.
6. FRIEDEN, B. R. (1967): Band-Unlimited Reconstruction of Optical Objects and Spectra, *Journal of the Optical Society of America*, Vol. 57 (8), pp. 1013–1019.

7. HELMSTROM, C. W. (1970): Resolvability of Objects from the Standpoint of Statistical Parameter Estimation, *Journal of the Optical Society of America*, Vol. 60 (5), pp. 659–666.
8. VAN DEN BOS, A. (1987): Optical Resolution: an Analysis Based on Catastrophe Theory, *Journal of the Optical Society of America*, Vol. 4, pp. 1402–1406.
9. VAN DEN BOS, A. (1992): Ultimate Resolution: a Mathematical Framework, *Ultramicroscopy*, Vol. 47, pp. 298–306.
10. VAN DEN BOS, A. – SWARTE, J. H. (1993a): Resolvability of the Parameters of Multiexponentials and Other Sum Models, *IEEE Transactions on Signal Processing*, Vol. 41 (1), pp. 313–322.
11. BORN, M. – WOLF, E. (1980): Principles of Optics, 6th edn., Pergamon, New York.
12. POSTON, T. – STEWART, I. N. (1978): Catastrophe Theory and Its Applications, Pitman, London.
13. VAN DEN BOS, A. (1993b): Critical Errors Associated with Parameter Resolvability, *Computer Physics Communications*, Vol. 76, pp. 184–190.

PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 41, NO. 3, PP. 185-200 (1997)

IMPROVED LOCALISATION FOR TRAFFIC FLOW CONTROL

Martin OSTERTAG

Institute for Industrial Information Systems
University of Karlsruhe, FRG
Hertzstraße 16, Bau 6.35
D-76187 Karlsruhe

Received: Nov. 22, 1995

Abstract

Localisation of vehicles plays an important role in future traffic control concepts. To solve this task several sensors may be used. Each of these different sensors has its own, specific disturbances. Incremental measurement of wheel rotation to get information about the covered distance is distorted by slip and rugged roads, measuring the orientation of the vehicle by means of a magnetometer suffers from internal and external disturbing magnetic fields, and vehicle vibrations disturb yaw rate measurement.

A method to improve autonomous localisation based on Kalman filtering is presented. By estimating the variance of the different sensor data the Kalman Filter parameters can be varied to achieve improved system behaviour. Results are presented for an in-town drive. The system is just about to be implemented in real-time in a test vehicle at the University of Karlsruhe.

Keywords: data-fusion, Kalman filter, autonomous localisation.

1. Introduction

Rising traffic creates the need for intelligent methods of traffic control. A basic requirement for intelligent control of traffic flow is the ability of a vehicle to navigate for a limited period of time without external aid. Thus the need for a method to estimate its position in respect to a known origin is obvious.

If possible, a localisation system should make use of sensors that can already be found in a modern vehicle to keep the costs and the complexity of the overall system low. In the system which is to be presented, the incremental sensors at the wheels which are used by the anti-lock braking system are chosen as basic sensors. To improve the quality of the localisation system, a magnetic field probe is applied as sensor for the terrestrial magnetic field. Additionally, data from a yaw rate sensor are available.

Yaw rate sensors will be used not only by the navigation system, but also by future improved driving stability control systems.

Each of the sensors has its own specific disturbances. v.d. HARDT, (1992) showed that Kalman filtering is a suitable method for fusing data provided by the different sensors. Modelling the sensor errors in combination with a Kalman filter approach improves the system performance significantly compared to the simple use of each sensor information alone.

2. Position Determination

The position of a vehicle in respect to a known origin O is definitely determined by the vehicle's x- and y-co-ordinates and its orientation Θ (*Fig. 1.*)

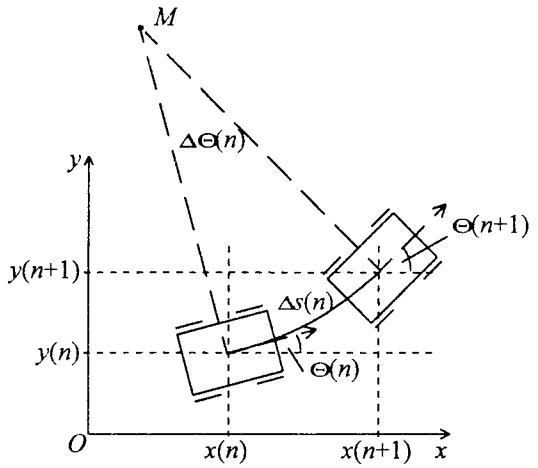


Fig. 1. Position of a vehicle

If the position of a vehicle is to be determined at time $t_n = t_0 + nT$, it is the task of a localisation system to determine the position at time $t_{n+1}[x(n+1), y(n+1), \Theta(n+1)]$ based on the (known) position at time $t_n[x(n), y(n), \Theta(n)]$ using the sensor data and applicable data-fusion methods.

2.1 Assumptions and Approximations

To calculate the position $\underline{x}(n) = [x(n), y(n), \Theta(n)]$ of the vehicle it is assumed, that during each localisation cycle T the vehicle moves with a con-

stant speed on a circle with a constant diameter. Centre of this circle is the instantaneous pole M (*Fig. 1*). If the localisation cycle T is sufficiently short, this assumption can be considered fulfilled.

To determine the new position from the covered distance $\Delta s(n)$ and the angular orientation increment $\Delta\Theta(n)$ during one localisation cycle, the arc $\Delta s(n)$ is approximated by the straight line $\Delta h(n)$ (*Fig. 2*). This approximation is based on the assumption that the angular increment $\Delta\Theta$ is not too big. The approximation error is 1.1% for an angular increment of 30° . Under normal conditions the angular increment is much smaller, so the approximation error is acceptably small.

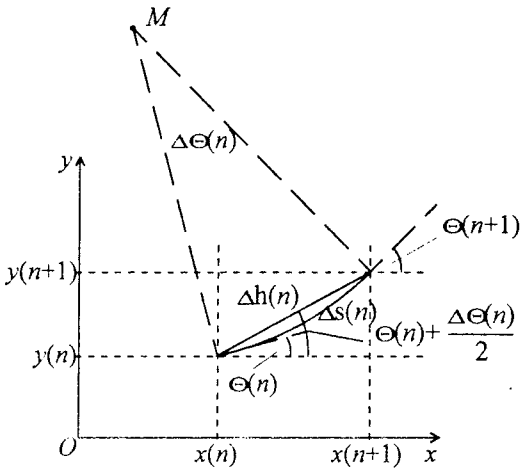


Fig. 2. Approximation of $\Delta s(n)$

2.2 Determination of the Vehicle's Position

Under the assumption and approximations made above, the following three recursive equations for determination of the position can be found:

$$x(n+1) = x(n) + \Delta s(n) \cdot \cos\left(\Theta(n) + \frac{\Delta\Theta(n)}{2}\right), \tag{1a}$$

$$y(n+1) = y(n) + \Delta s(n) \cdot \sin\left(\Theta(n) + \frac{\Delta\Theta(n)}{2}\right), \tag{1b}$$

$$\Theta(n+1) = \Theta(n) + \Delta\Theta(n). \tag{1c}$$

The angular increment $\Delta\Theta(n)$ as well as the orientation $\Theta(n)$ appear in every equation, in *Eq.* (1a) and (1b), which determine the new x- and y-co-

ordinates, non linearly connected with the covered distance $\Delta s(n)$. Thus it is especially important to determine the orientation of the vehicle and the angular increment as exactly as possible.

Under the condition of exactly known geometric properties of the wheels and the axle, it is possible to determine the covered distance of the vehicle $\Delta s(n)$ as well as the angular increment $\Delta\Theta(n)$ solely from the distances covered by two wheels of a single axle:

$$\Delta s(n) = \frac{\Delta s_r + \Delta s_l}{2}, \tag{2a}$$

$$\Delta\Theta(n) = \frac{\Delta s_r - \Delta s_l}{L}. \tag{2b}$$

Δs_r and Δs_l are the distances covered by the right respectively left wheel during the localisation period, L is the effective tread.

The covered distance is readily determined by this method, while the determination of the orientation, i.e. the angular increment, is not sufficiently possible. Especially the integration process in *Eq.* (1c) augments errors due to the use of incremental information only. The reasons for this are explained in chapter 3.2.1.

The final localisation system uses the average of the covered wheel distances to determine the covered distance of the vehicle, while the orientation of the vehicle is determined using data fusion methods explained in chapter 4 for the data of all sensors that provide information on the angular increment and/or the orientation.

3. Used Sensing Devices

Sensors used for localisation tasks can be divided in two classes. Proprioceptive sensors provide differential information on physical quantities, i.e. a value is measured in relation to the vehicle's position after the last localisation cycle. Proprioceptive sensors are incremental shaft encoders, accelerometers or yaw rate sensors. Exteroceptive sensors measure a physical quantity, e.g. the orientation of a vehicle, absolutely in relation to an external reference. Examples are a compass, triangulation methods or cameras mounted on the vehicle.

In the system which is dealt with here, two different kinds of proprioceptive sensors are used. Besides the incremental encoders of the anti-lock braking system a yaw rate sensor is mounted between the front seats. A compass of Förster type is used to measure the terrestrial magnetic field and is the only exteroceptive sensor for the vehicle's orientation.

3.1 Description of the Sensors

Measurement of the wheel speed is performed by inductive sensors which scan the teeth of a toothed wheel mounted on the axle of every wheel. The signals are conditioned using differential amplifiers and Schmitt triggers and then connected to counter ports of a microcontroller. The microcontroller latches the actual counter value for the first and last slope of the signal for each wheel during a localisation cycle as well as the number of slopes actually encountered for each wheel. The covered distance may be determined as:

$$\Delta s = 2 \cdot \pi \cdot R \cdot \frac{T \cdot f_C}{Z} \cdot \frac{(N_{Sl} - 1)}{(N_l - N_f)}. \tag{3}$$

Here, R is the radius of each wheel, N_{Sl} the number of slopes, Z the number of teeth on the toothed wheel, T the localisation cycle, f_C the frequency of the counter and N_f and N_l the counter values for the first and last slope. As the amplitude of the sensor signals decreases with decreasing speed, the suitability of the wheel sensors is limited to speeds above 5 km/h.

To measure the yaw rate, which is the rotational speed of the vehicle chassis in regard to the vertical axis, a vibratory gyroscope (MURATA, 1990) is applied. The measurement effect is based on the measurement of Coriolis force that detunes a vibrating, triangular bar.

The magnetic field probe is a Förster sensor that consists of two coils which are arranged with an angle of 90° to each other. They are coiled around a cross-shaped ferro-magnetic former. So it is possible to measure the projection of the terrestrial magnetic field into the horizontal plane. A ramp-shaped current drives the coils from negative saturation to positive saturation, the saturation region is measured by a superimposed, high-frequency AC-current. Superimposed magnetic fields yield in a shifting of the saturation region, which serves to measure the superimposed magnetic field.

3.2 Errors of the Proprioceptive Sensors

3.2.1 Incremental Wheel Sensors

The use of wheel sensors to measure the covered distance and the angular increment of the orientation is corrupted by systematic and stochastic errors. Systematic errors arise mainly from geometric inaccuracies. The wheel diameters vary with the vehicle speed, the tire pressure, temperature and the vehicle load. The effective tread L (*Eq.* (2b)) varies mainly with the speed and the curve radius. This effect is especially big for the front

wheels, because here the effective tread depends, caused by the mechanical construction, strongly from the angle of turn and the wheel suspension. Stochastic errors are caused by variable slip, road inclination, rough road etc. Partly, the systematic errors can be minimized by calibration runs, but the stochastic errors can hardly be detected. Because of this, the measurement errors of the angular increment $\Delta\Theta$ using wheel sensors are modelled as being a constant which is used for the Kalman calculations. Additionally, plausibility checks help to avoid mistakes during anti-lock braking or during situations with extremely big slip. At speeds below about 10 kilometres per hour, the angular information from the wheel sensors is ignored due to the sensor problems at low speed (cf. section 3.1).

3.2.2 Vibratory Gyroscope

The gyroscope suffers from two different errors. First, there has to be considered an offset drift, which is varying slowly with the temperature. This drift is compensated using a long-term highpass filter. The second error source are higher frequency disturbances which are mainly caused by vibrations of the vehicle chassis. To get a measure for these errors, the signal from the gyroscope is oversampled and a second-order polynomial is interpolated using the standard least-squares algorithm. The interpolation error is used to determine the variance $\sigma^2_{\Delta\Theta,GS}$ of the yaw rate error. As the variance is non-zero even when the interpolation error disappears, the variance is determined as shown in *Fig. 3.* A minimum variance is chosen for a disappearing interpolation error, and for the highest observed error a maximum variance is chosen.

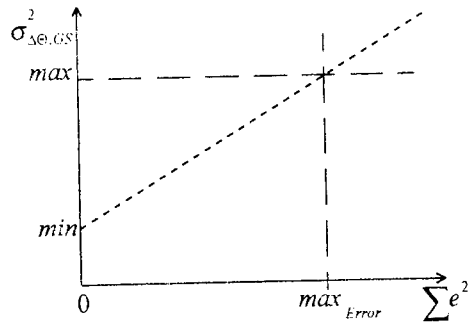


Fig. 3. Determination of yaw-rate error variance

3.3 Magnetic Field Probe

Error detection and error compensation for the magnetic field probe as the only exteroceptive sensor in the system requires special attention. The errors can be separated into static and dynamic errors as well as into orientation-dependent and orientation-independent errors.

3.3.1 Sources of Error

The main sources of error for the compass can be found in magnetic fields that are superposed to the terrestrian field. Internal interference fields are those fields that have their origin from within the vehicle. They can be subdivided into orientation-dependent and -independent fields. Orientation-independent fields arise from permanent magnetic parts of the vehicle chassis or from on-board DC loops caused by consumers like a rear window defroster. They yield a shifting of the origin for the compass. Orientation-dependent interference fields are caused by poles induced in magnetically soft parts of the chassis and effect a distortion of the circle expected for the compass when turning the vehicle to an elliptical curve. Together the static fields result in a situation as shown in *Fig. 4*. As long as there are no changes made in the chassis (turning consumers on and off are such changes!) the above described static fields interfere with the terrestrial magnetic field and can be measured and compensated by calibration runs.

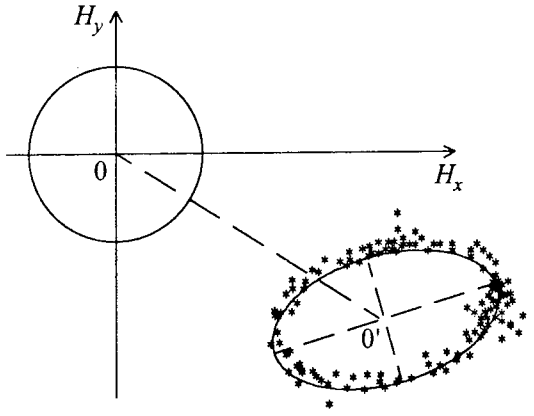


Fig. 4. Shifting and distortion of the magnetic field probe signal

Additionally, dynamic fields are interfered, caused either by the vehicle itself (consumers, sliding sun roofs) or interfered from the outside (tramway

overhead contact lines, trucks that are overtaken). To detect and compensate errors caused by these fields it is assumed that the dynamic errors do not cause distortion but only shifting of the magnetic curve in *Fig. 4*. The errors caused by the vehicle itself can be minimised by a suited mounting position for the compass and by detection of on-board reasons like the rear win- dow defroster or the sliding roof and compensation by additive calibration drives. Besides of the errors caused by interfering fields errors arise from the sensor construc- tion itself. The magnetic field is measured in two horizontal directions only. Measuring the horizontal projection of the terrestrial magnetic field causes errors when the vehicle moves upward or downward, if the road is inclined or if the vehicle is asymmetrically loaded. These errors can not be detected with the used sensors.

3.3.2. Error Detection and Compensation

The compass generates 24 measurement values during every localisation cycle. During a calibration drive the parameters of the shifted ellipse are determined, so that the measured values can be re-transformed to the unit circle. Considering the mounting angle to the chassis length-axis and the magnetic declination, it is possible to determine the orientation of the vehicle straightforward.

For validation of the measured values a ring-shaped and a sector-shaped filter are used. After transformation to the unit circle the distance of the values to the origin is calculated. If the distance is significantly different from the expected, the values are considered invalid (ring-shaped filter). Equally, the deviation of the orientation values Θ_i is judged, and values with a deviation from the mean $\overline{\Theta}$ higher more than a predetermined angle α_{filter} are considered invalid. The valid measurement values (*Fig. 5*) are then used to determine the orientation Θ_{MFP} as mean of the valid values Θ_i.

The hardest problem to be dealt with when using the compass is the compensation of the ellipse shifting caused by dynamic disturbances. Here, all available values from the field probe are used. The shifting of the origin is performed, when the measured values cover a angular region sufficiently large, the number of measured values is high enough and if the calculated value for the shifting is constant for several localisation cycles. If these assumptions are true, then the shifting is performed by interpolating a unit circle using the sensor data and the shifting parameters are updated.

To determine the variance of the measurements of the magnetic field probe, the mean distance of the measured and re-transformed values to the unit circle is taken into account. The principle is the same as described

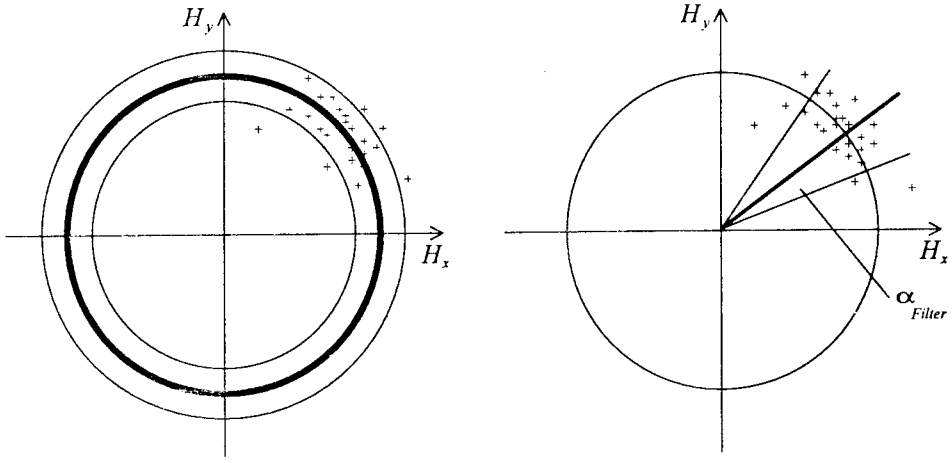


Fig. 5. Ring- and Sector-Shaped Filter for Validation of the Compass Values

for the yaw rate sensor (section 3.2.2). Test drives showed, that instead of a linear function between the two fixed points of *Fig. 3*, a square root function yields better results.

4. Determination of the Orientation using Kalman Filters

4.1. Signal Model to Determine the Orientation

To determine an optimal estimation for the orientation of the vehicle using the redun- dant information provided by the different sensors, the following signal model is used as a base for the filtering process:

$$\begin{bmatrix} \Delta\Theta(n+1) \\ \Theta(n+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta\Theta(n) \\ \Theta(n) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot u(n) \tag{4}$$

Θ, the signal vector, is composed of the two variables $\Delta\Theta$ and $\Theta \cdot u(n)$ is the (unknown) value of the angular acceleration at instant n. $u(n)$ is modelled as white noise with mean zero and variance σ_u^2. $\Delta\Theta(n)$ and $\Theta(n)$ the output variables of the system that are corrupted by noise depending on the used sensors:

$$\begin{bmatrix} \Delta\Theta_m(n) \\ \Theta_m(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta\Theta(n) \\ \Theta(n) \end{bmatrix} + \begin{bmatrix} e_{\Delta\Theta}(n) \\ e_{\Theta}(n) \end{bmatrix} \tag{5}$$

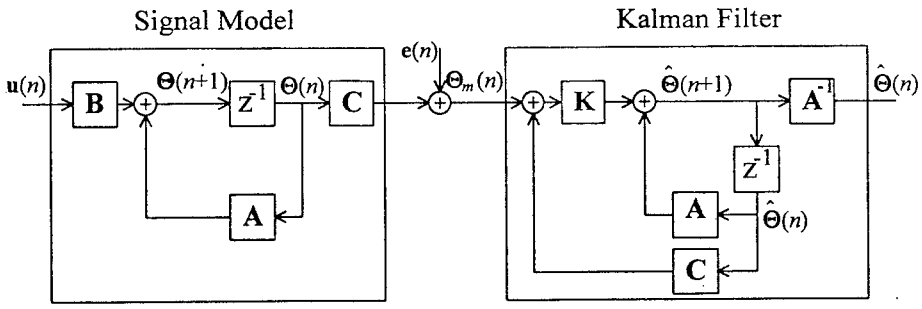


Fig. 6. Signal Model and Kalman Filter Structure

The output noise $e_{\Delta\Theta}(n)$ and $e_{\Theta}(n)$ is modelled as gaussian noise with mean zero that is statistically independent. The variance matrix $\mathbf{V}_e$ of the noise for the different sensor types is determined as described in chapter 3.

Fig. 6 shows the signal model and the Kalman filter structure. A prediction for one step is performed, which is then used to estimate the optimal values of $\Delta\Theta(n)$ and $\Theta(n)$ (KRONMÜLLER 1992, and KROSCHEL 1988).

4.2. Recursive Evaluation of the System

Using the matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{K} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \tag{6}$$

for the evaluation of the system shown in *Fig. 6*, the estimated value for the orientation and angular increment at instant n is:

$$\hat{\Theta}(n+1) = (\mathbf{A} - \mathbf{K}(n) \cdot \hat{\Theta}(n) + \mathbf{K}(n) \cdot \Theta_m(n) \tag{7}$$

with the recursive equations

$$\mathbf{K}(n) = \mathbf{A}\mathbf{V}_k(n)\mathbf{C}^T[\mathbf{V}_e(n) + \mathbf{C}\mathbf{V}_K(n)\mathbf{C}^T]^{-1} \tag{8a}$$

$$\mathbf{V}_K(n+1) = (\mathbf{A} - \mathbf{K}(n)\mathbf{C})\mathbf{V}_K(n)(\mathbf{A} - \mathbf{K}(n)\mathbf{C})^T + \\ + \mathbf{K}(n)\mathbf{V}_e(n)\mathbf{K}^T(n) + \mathbf{B}\sigma_u^2\mathbf{B}^T \; . \tag{8b}$$

Here, $\mathbf{V}_K(n+1)$ is the covariance of the prediction and $\mathbf{K}(n)$ is the filter gain. The elements K_{ij} of the filter gain matrix are:

$$\begin{aligned}
K_{11}(n) &= V_{K,11} \cdot V_{K,22} - V_{K,12} \cdot V_{K,21} + V_{K,11} \cdot \sigma_\theta^2(n) \\
K_{12}(n) &= V_{K,12} \cdot \sigma_{\Delta\Theta}^2(n) \\
K_{21}(n) &= V_{K,11} \cdot V_{K,22} - V_{K,12} \cdot V_{K,21} + (V_{K,11} + V_{H,21}) \cdot \sigma_\theta^2(n) \\
K_{22}(n) &= V_{K,11} \cdot V_{K,22} - V_{K,12} \cdot V_{K,21} + (V_{K,12} + V_{H,22}) \cdot \sigma_{\Delta\theta}^2(n) \qquad (9)
\end{aligned}$$

$V_{K,ij}$ are the elements of the prediction covariance matrix $\mathbf{V}_K(n)$, $\sigma_\Theta^2(n)$ the variance of the noise of the orientation measurement and $\sigma_{\Delta\Theta}^2(n)$ the variance of the angular increment noise.

4.3. Cascading of the Kalman Filters

Using all three sensors that provide information on the orientation or the angular increment a Kalman filter structure is shown in *Fig. 7*. The first Kalman filter (KF1) fuses the information from the magnetic field probe and the gyroscope to obtain an improved estimate for the vehicle's orientation. The following, second Kalman filter (KF2) uses the information provided by the wheel sensors. The value for Θ_m at the input of KF2 is the output of KF1. The corresponding variance for the orientation measurement is $V_{K,22}$, the prediction variance of KF1.

In case of a sensor defect, for a first step the information provided by that sensor is kept from the prior period and the sensor variance is increased. If the sensor defect lasts for several localisation periods, the structure of the cascaded Kalman filters is changed. If the missing information is either the one from the gyroscope or the one from the wheel sensors, the system keeps operating with a single Kalman filter as described in section 4.2. If the defect sensor is the compass, the Kalman filter structure has to be changed. The absolute value for the orientation cannot be measured, so the absolute orientation is estimated by adding one of the incremental sensor values to the optimal value estimated one step in the past (*Fig. 8*).

Here, each of the input values for the Kalman filter algorithm is given a variance as determined for the incremental sensors.

5. Experimental Results

In the following section some of the experimental results are presented. First, values for the estimated variance limits that led to a satisfactory

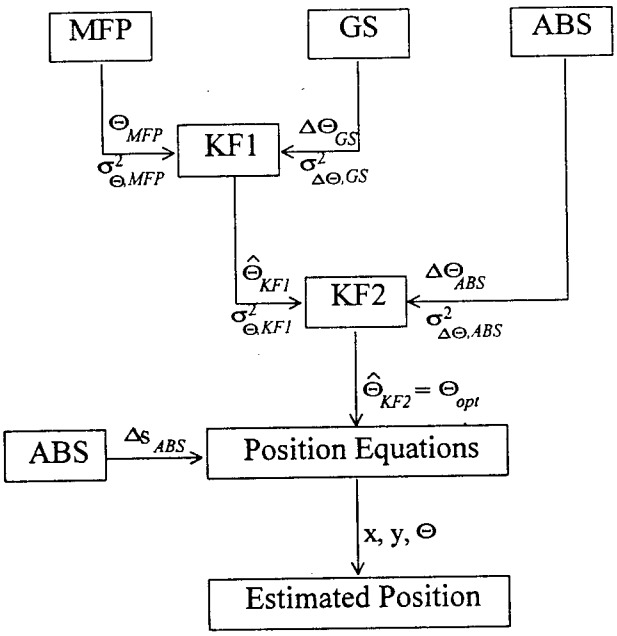


Fig. 7. Signal Flow for Data Fusion

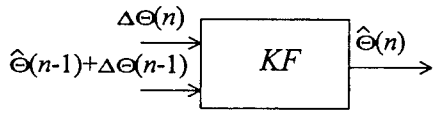


Fig. 8. Reduced Kalman filter structure with compass defect

system behaviour are presented. Thereafter, the new algorithm is compared to the simple combination of the sensor values for different test drives.

5.1. Choosing the Variance and Algorithm Parameters

The geometrical properties of the test vehicle were determined during several calibration drives. These include straight drives to determine the wheel

diameters under dynamic circumstances as well as several cycling drives to determine the tread.

During the development and test it showed, that the sensor variance region is the most important tool to tune the algorithm. For the wheel sensors, a constant variance $\sigma^2_{\Delta\Theta,ABS}$ of $(0.5^\circ$ per system cycle$)^2$ was chosen. In general, yaw rate measurement results in a better reliability of the data, so the minimum value $\sigma^2_{\Delta\Theta,GS}$ for the gyroscope (cf. *Fig. 3*) was chosen to be $(1^\circ$ per system cycle$)^2$ and the maximum value is $(2^\circ$ per system cycle$)^2$ for the maximum interpolation error observed during test drives.

When choosing the variance estimation for the magnetic field probe, it showed that a relatively high variance even for undisturbed measurements improves system behaviour. The reason is that the compass as the only exteroceptive sensor has a high long-term influence on the orientation estimation. Dynamic disturbances, which last for only a few system cycles, can easily be suppressed by choosing the compass minimum variance of $(10^\circ)^2$. Here, the maximum variance was chosen as $(50^\circ)^2$.

The variance of the unknown system input noise, the angular acceleration, was chosen to be fixed as $(10^\circ$ per $($system cycle$)^2)^2$. For further investigations, a functional link between the vehicle speed and the system input variance might provide even better results.

5.2 In-Town Test Drive

The test drive presented here is a track in Karlsruhe. *Fig. 9* shows the results for the Kalman filter algorithm compared to algorithms that use only the compass data or only the gyroscope data for determination of the vehicle's orientation. Starting point is at the coordinate origin in the lower right corner of *Fig. 9*. The black asterixes are reference points taken from a map, after every 100 localisation cycles a cross is printed on the respective trajectory. The Kalman filter algorithm reaches the target after a covered distance of 6.5 km with an accuracy of 3.5%, the algorithm using the compass shows an error of twice the size, and the results of the algorithm that uses the gyroscope data only are not satisfactory at all.

It can be seen that the algorithms using the compass are better in keeping the orientation than the algorithm using incremental information only. Orientation is lost right at the beginning of the drive, afterwards the vehicle is steadily turning too far to the right.

Reasons for the superior performance of the Kalman algorithm in comparison to the simple use of the compass data can be explained using *Fig. 10*. The orientation information of the compass alone is plotted as a dashed line, while the orientation for the Kalman algorithm is plotted as

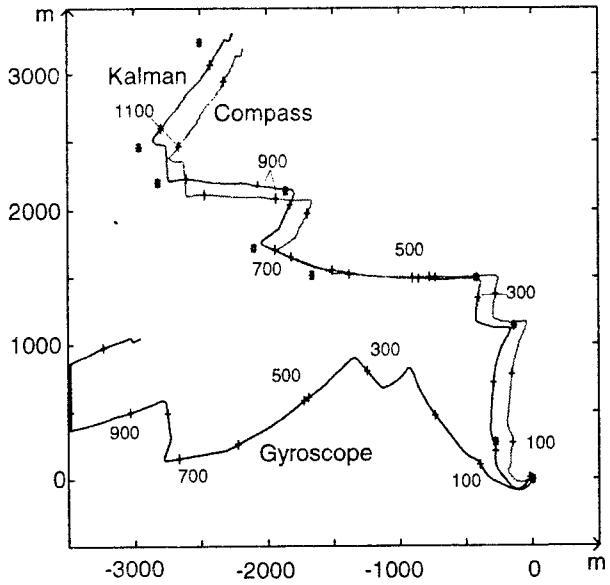


Fig. 9. In-town test Drive

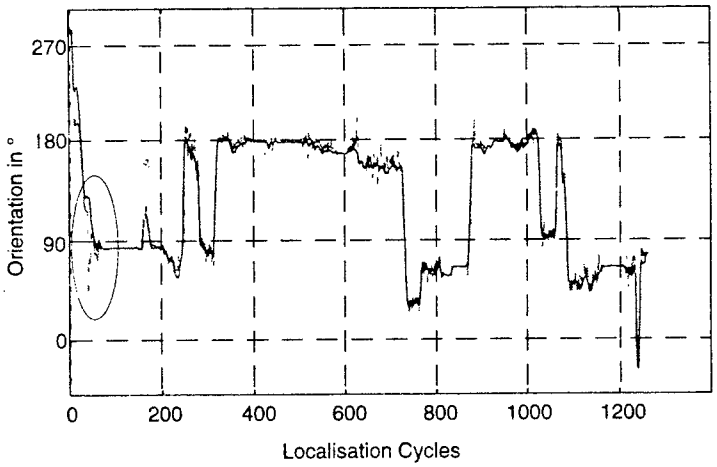


Fig. 10. Kalman filter and compass orientation for test drive

solid line. The angle information of the compass suffers from short-time disturbances. At the beginning of the track, the road crossed railway lines close to the central station of Karlsruhe. This leads to massive disturbances

in the compass data. The orientation determined by the Kalman algorithm does not follow the compass data because of the use of additional data from the incremental sensors.

More test drives show similar results. Still it shows that preprocessing of the sensor data is most important to achieve satisfactory results. If, for example, the medium and long-term errors of the magnetic field probe are not compensated properly, the estimated orientation has a bias of several degrees. This leads to significant localisation errors.

6. Conclusions

A method for autonomous localisation of vehicles based on wheel rotation measurement, yaw rate measurement and a compass was presented. The determination of the covered distance is performed based solely on the wheel information, while the orientation of the vehicle is determined based on all available sensors. Combination of the sensor data is performed using cascaded Kalman filters. Modelling of the sensor errors gives the possibility for adaptive variation of the noise variances used to calculate the Kalman filter gain. So, an improved estimation for the vehicles orientation can be achieved.

A careful pre-processing of the sensor data is very important. Special care must be taken considering the compensation of the offset drift of the compass. Badly compensated compass data deteriorates the function of the localisation system remarkably.

Development of algorithms to improve estimation of the covered distance are subject of further investigations as well as map-matching methods. This will not only provide the opportunity for global localisation but also the possibility to adopt parameters which are subject to change during operation of the vehicle. Mainly, that are the wheel diameters and the offset drift of the gyroscope. Methods to improve compass compensation are also investigated.

The localisation system presented is implemented in real-time in a test vehicle at the Institute for Industrial Information Systems at the University of Karlsruhe.

References

1. ABIDI, M. A. – GONZALEZ, R. C. [ed.] (1992): Data Fusion in Robotics and Machine Intelligence, Academic Press, San Diego
2. BRAND, A. (1994): Lokalisierung eines Fahrzeugs durch Multisensor-Datenfusion, Institute for Industrial Information Systems, University of Karlsruhe. (In German)

3. KRONMÜLLER, H. (1991): Digitale Signalverarbeitung, Springer Verlag, Heidelberg. (In German)
4. KROSCHEL, K. (1988): Statistische Nachrichtentheorie, Zweiter Teil: Signalschtzung, Springer Verlag, Heidelberg. (In German)
5. MURATA (1990): Data Sheet to Gyrostar ENV-05S
6. V.D. HARDT, H. J. (1992): Lokalisierung eines mobilen Roboters mit Hilfe von Wegmessung und Erdmagnetfeldmessung, Institute for Process Measurement and Automation, University of Karlsruhe. (In German)

PERIODICA POLYTECHNICA SER. ELECTR. ENG. VOL. 41, NO. 3, PP. 201-230 (1997)

THE RAFAEL MULTI-TARGET HETEROGENEOUS SIGNAL-FLOW GRAPH COMPILER

Gábor PALLER and Klára CSÉFALVAY

Department of Electromagnetic Theory
Technical University of Budapest
H–1521 Budapest, Hungary,
e-mail: paller@evt.bme.hu
csefalvay@evt.bme.hu

Received: June 23, 1995

Abstract

This paper describes a signal-flow graph compiler which produces distributed code for heterogeneous target systems. The compiler is devoted for mainly Digital Signal Processing problems. The code generator features reprogrammable operation library, the static scheduler supports fully heterogeneous systems and the input graph may contain run-time decisions in a limited way. The system has been implemented on IBM PC compatibles under MS-Windows so it does not require expansive host computer.

Keywords: compile-time scheduling, parallel processing, heterogeneous architectures.

1. Introduction

Writing programs for the modern Digital Signal Processors (DSPs) introduce difficult tasks for the software engineers because a painful trade-off exists between the computing power and the productivity/task complexity. Unfortunately the existing and well-known higher level programming environments (for example the 'C' language) perform very poorly on the DSP platforms because being general languages they cannot exploit the special capabilities of the DSPs (circular buffers, parallel instructions and so on) or avoiding pipeline effects. This can cause extremely high performance loss (can be as much as 1000% compared to the assembly realization). Several developments were made to improve C compilers on DSP platforms (LEARY and WADDINGTON, 1990) but generally they use system or DSP dependent language extensions and their performance is still not really convincing. So the developers have to choose — writing the DSP code in assembly for achieving higher performance thus lower hardware cost or using a high-level environment which will speed up the development but decrease the efficiency of the DSP so that more expensive DSPs must be chosen. It can even happen that the problem cannot be solved on high level.

The other problem is the embarrassing abundance of DSP architectures and languages. One often faces the problem of porting existing results

onto other DSP platforms. If the code is written in assembly, this will be a long and tiresome process. Some 'common languages' are needed but not having efficiently realizable high level platform this solution does not seem to be promising. Nowadays the solution is sought toward optimized software libraries (like the SPOX) which try to combine the power of assembly routines with the efficiency of C. The SPOX does accelerate the developing process but it is a fixed set of routines and if we extend it (for example we need an arithmetic routine or new algorithm that the SPOX cannot offer) we still have to write it in assembly losing the portability.

Nowadays the parallel DSP is in the focus of attention, first of all because real-world DSP problems often require immense computing power. A number of existing DSPs can be used for parallel realizations, some of them has been designed especially for parallel computing for example Texas Instrument's TMS320C40, TMS320C80 and Analog Devices ADSP21060. The task scheduling is an important part of the multiprocessor implementation of DSP algorithms. This equally means partitioning the tasks among multiple DSPs and scheduling the tasks on each DSP. Generally parallel programs are scheduled 'by hand' in the existing parallel development systems which is a difficult task and in the case of more complex tasks it cannot be done effectively. The other approach used frequently in the existing DSP operating systems uses the well proven real-time operating systems scheme (sometimes time-sliced scheduling is added). This scheme is based on separate tasks and a task scheduler program which changes the tasks when it is necessary. This task scheduler requires processing time.

Speciality of the DSP algorithm is that it does not require much run-time decisions. Very handy description form of these algorithms is the *signal-flow graph (SFG)*. Signal-flow graph is a graphical description of an algorithm in which computations are represented by graph nodes and dependencies among the computations by graph branches. If we can cluster enough nodes together that their dependency graph and execution time do not depend on the input values, we can schedule in *compile time* thus eliminating the processor load of the dynamic scheduler.

Thus the DSP code generation problem is the following: we need a system which is flexible enough to be adapted to several existing DSP platforms, avoids the power loss of the high-level languages, solves the partitioning and scheduling problems and in addition it is easy-to-use for the DSP algorithm developer who is generally not a programmer. A proposition for this problem will be presented in this document describing Rafael, an intelligent code generator based on signal-flow graphs.

Rafael was designed as a small, flexible system which can run even on very small computers (it is implemented under Microsoft Windows on IBM PC compatible computers). It is a SFG compiler integrated into a simple

framework which allows DSP algorithms to be described in SFG form and the compiler translates this description into program for a heterogeneous multiprocessor hardware. The compiler distributes the SFG on the multiprocessor system, schedules the operations on each processor, creates the communication scheme among the processors and generates executable assembly source program for each processor. Rafael features a programmable DSP database and code generator library so it can be adapted easily to any processor. Small resources of the host computer do not allow us to compete with the comprehensive features of existing SFG compilers hosted on workstations but we hope to prove that Rafael can compete successfully on several domains with those systems.

2. Existing Data-Flow Compilers

A number of block-diagram based design systems have been introduced in the literature. We mention here the commercially available DSPlay (Burr-Brown) and SPW (Signal Processing Workstation) (Comdisco) systems. DSPlay is PC-based, it can simulate the input block-diagram and can generate code for AT&T DSP32. The Comdisco system started as a simple simulator but actually it is able to produce highly optimized code for almost all the DSP types and can even generate circuit description. Since June 1994 the partitioning on multiprocessor DSP system must have been done by hand. The Cathedral system (DE MAN et al., 1986; LANNEER, 1993) devoted to circuit synthesis features SFG partitioning-scheduling but it uses the Silage functional language (GENIN et al., 1990) as its input. The Ptolemy system (BUCK et al., 1991; BUCK, 1993; BUCK et al., 1994) is the most comprehensive existing simulation/code generation system. Ptolemy supports the coexistence of different computation models (called *domains* by their terminology) and offers clearly defined object-oriented interface for defining a new domain. Existing domains include static dataflow (LEE – MESSERSCHMITT, 1987), dynamic dataflow (BUCK, 1993), discrete event, message queue and communicating process (BUCK et al., 1994) models. Ptolemy makes almost no assumption about the internal structure of the computation models it supports, it is the biggest strongness and weakness of this system. It is a strongness as it allows modelling the whole system including its software, hardware and communication parts in one framework. It is weakness as Ptolemy allows mixing computation models that do not coexist well, it does not force a good design style. Nevertheless, Ptolemy has huge impact on the field and its importance grows continuously as existing computation models and tools are integrated with it.

Many ideas of the structure of Rafael were borrowed from the now historical Gabriel system. Gabriel was phased out in favor of the much bigger Ptolemy system but we found that some solutions introduced in Gabriel fit well to our much less powerful target platform. Gabriel (LEE et al,. 1989) was the first system capable of generating executable code at Berkeley in which the synchronous dataflow paradigm was implemented. Its predecessor, BLOSIM (MESSERSCHMITT, 1984) was only a simulator.

The operations (or *actors* by the terminology of the Berkeley team) are called *stars.* A cluster of stars forming an interconnected SFG is called *galaxy.* The final SFG can be hierarchical composed of a number of galaxies, a set of interconnected galaxies is called *universe.* Gabriel has two levels of user interface. The graphical dataflow organization is used where appropriate: when describing the algorithm in dataflow format. The stars have textual definition. This mixed description form helps to avoid the common problem of the graphical description systems which use graphical terms where they are not handy.

One of the most striking features of Gabriel is its *programmable star library* which influenced a lot the database of our Rafael system. A Gabriel star is described by a Lisp structure. The star library entry has a *header* and a *function body.* The header structure stores information about the inputs and outputs of the operation, a short textual description for human readers and the parameters and their default values. An entry in the header points to the *star function* which gets executed whenever the star is invoked. This star function can actually execute the operation assigned with the star in simulation mode or can generate a code for the actual target processor in code generation mode. It is important to note that the code generator star library is written in Lisp so a code generator function can be quite intelligent when it decides on the text to be generated depending on the parameters. size of the inputs, etc. Beside the star function, a Gabriel star can have initialization/termination functions that are called once before the first invocation and after the last invocation of a star. Processors are described in a similar way creating Lisp lists that contain the target system characteristics: number of processors, processor memory, special hardware units connected to processors, communication channel characteristics between the processors and communication code generator routines. The Gabriel system is strictly homogeneous: there can be only one star library in the memory.

The Gabriel system has the following interesting features:

- It handles multiple sample rates which result naturally from its input format, the synchronous dataflow graph.

- It has a second user level, the star library programming level in Lisp which allows the user to create new stars easily and to add intelligent optimization/code generation features to the existing star library.

The main weaknesses:

- It does not address the question of data dependent constructs, if-then-else, case, etc.
- It does not support heterogeneous systems.
- Its scheduler cannot be considered efficient.

Another system that influenced greatly our work is SynDEx (SOREL, 1994). SynDEx is a code generator environment designed, to be interfaced with the synchronous language compilers, SIGNAL (LE GUERNIC et al., 1991), LUSTRE (HALBWACHS et al., 1991), ESTEREL (BOUSSINOT – SIMONE, 1991). It has a graphical and textual user interface that allows users to construct the algorithm block diagram entirely in SynDEx. It is designed, however, rather to receive the algorithm graph from a synchronous language compiler. Actually SynDEx is interfaced in such a way with SIGNAL (BOURNAI, 1994) and work is under way to create a common format for the SIGNAL, LUSTRE, ESTEREL languages so that they can send the result of compilation to SynDEx or other code generators. The algorithm model of SynDEx is the *conditioned signal-flow graph.* It means that each node has a clock it is associated to which results in a *condition input* for each node (*Fig. 1*).

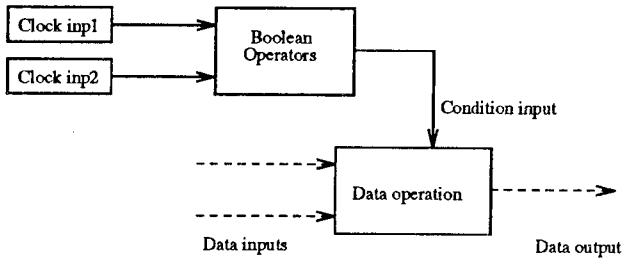


Fig. 1. Conditioned signal-flow graph

A node is fired if all its input variables (including the control variable) have been produced by predecessor nodes and its control variable is *true.* The scheduler considers the condition input dependency as any other dependency: it is equivalent with supposing that each condition is true and each node can be executed. This way the original conditioned signal-flow graph is transformed to a synchronous signal-flow graph and static scheduling can

be used. The original conditioned signal-flow graph is thus partitioned into a condition calculating part (which is unconditioned) and a data processing part (which can be conditioned). Is is the responsibility of the SIGNAL compiler (or the input graph designer) that a proper condition signal be assigned to each node.

The biggest problem about the SynDEx system is caused by the way it handles the conditions. The actual implementation does not use the condition tree (AMAGBEGNON et al., 1994), constructed laboriously by the SIGNAL compiler, the hierarchy of clocks disappears, all the clocks become 'level 1' clocks (inserted just under the root clock). The code generator does not group operations scheduled one after the other with the same conditions into one `if ... endif`. Other drawbacks are that SynDEx does not support heterogeneous architectures and it can generate only C code.

3. Major Design Considerations of the Rafael System

The Rafael structure was designed according to the four main goals introduced at the beginning of this chapter. The support of heterogeneous systems needed a flexible operation library or — even better — programmable code generator module. Considering the code generator programmer's convenience, compiled languages can be quickly eliminated because it would need the recompiling and relinking of the code generator modules each time the database is modified. A system constructed in this way would be much more prone to system crashes as compiled languages allow great liberty in manipulating the system resources. We decided that reprogrammable parts of the code generator be implemented in an *interactive, interpreted language*. As we intended to provide the possibility of important intelligence in these modules (as they determine the quality of the code generated) we wanted to choose a more powerful language. Considering the possible candidates we chose Lisp because of the following advantages:

- It is a very powerful language that allows run-time program creation and it is equipped with efficient database handling capabilities.
- Lisp interpreters are available in relatively small memory requirement versions which fit well to the small computer (PC) we planned the system to run on.
- Excellent quality public domain versions have been written and distributed for several platforms in source code.
- It is a common language in CAD systems.

We must consider, however, the slow execution speed of Lisp whichis an even more serious obstacle on a small PC system. Although in the sense

of ease of programming it would have been more advantageous to realize the system entirely in Lisp, this solution would have resulted in unacceptable run time on the target system.

4. The Structure of the Rafael System

For the reasons mentioned in the previous section reason we choose a hybrid structure depicted in *Fig. 2.*

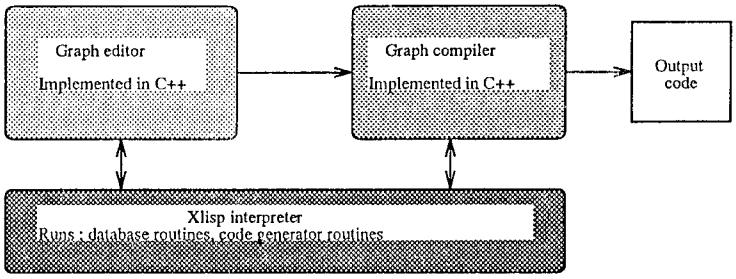


Fig. 2. Structure of the Rafael software

Each part of the software where user modifications are not supposed was implemented in C^{++}. This gives us a relatively powerful language with acceptable execution speed. Programmability is provided at Lisp level where an interface has been defined for the database and code generator programmer. By means of this interface the user can extend the database and the code generator library. The compiler core calls these routines from C^{++} level and uses their return value appropriately.

This solution needed separate tasks and interprocess communication between the tasks. The minimal 'operating system' that is sufficiently popular and needs small resources was the Microsoft Windows. At that time Linux (a small Unix version for PCs) was not in the state that we could have considered it as an alternative against Windows. By my personal opinion Windows is a poorly designed, inefficient 'operating system', today we would choose some other platform.

Thus, Rafael was implemented under MS-Windows, parts of this software (*Fig. 2*) run as separate Windows tasks and they are connected through the interprocess communication channels of Windows. The popular Xlisp was chosen as Lisp interpreter for Rafael because it is close to Common Lisp and it is available in C source. Xlisp was ported to Windows platform and the necessary interprocess routines were inserted that allows this Lisp interpreter to run as a server task.

The three Rafael software components have the following tasks.

Graph editor The name is a bit exaggerating as the Rafael framework is far from a comfortable working environment. It features a multi-screen text editor for creating/modifying graphs in textual format, initializes the Xlisp server and launches the Rafael compiler on the actually edited graph.

Graph compiler It is the SFG compiler. The program analyses graph description, makes the scheduling and generates the output text. It can run standalone as well, not only from the framework.

Lisp interpreter The operation database and its associated code generator routines are realized in Lisp. The client programs launch the server and send requests to it through interprocess links. Requests are actually Lisp commands which are executed by the server and the result of the Lisp command evaluation is returned to the caller C^{++} program.

As we can see the Rafael software architecture is very similar to that of Gabriel hence the similarity of the names. Rafael is different from Gabriel at the following points:

- Rafael's whole structure is adapted to the small host systems it runs on. Not the whole compiler was implemented in Lisp, only a part of it.
- As we will see, Rafael's whole design including the database, the scheduler it uses is adapted to heterogeneous systems. Gabriel was *multi-target* as it supported multiple start libraries. Rafael is *truly heterogeneous* as multiple target processors can coexist in the same operation library.
- Rafael supports a limited form of run-time decisions as its importance has been underlined many times both in the literature and in the practical engineering work. It will be detailed in section 6.
- Rafael features more advanced and efficient scheduler algorithms.

5. Rafael Nodes and Connections

The Rafael software model defines *nodes* that represent certain operations and *connections* between them. Nodes can be of the following types.

Operations Operations cover functions attached to a certain node. An operation is a parametrizable function. The number of inputs, outputs, the execution time and the operation of the function itself can depend on constant parameters.

Probes Probes cover functions whose task is to acquire input data from the environment of the dataflow system and send output data to the environment of the dataflow system. Probes are treated as simple

operations (with non-zero execution time, if necessary), the only difference is that they are explicitly forced to certain processors by the user. It derives from the fact that in a given hardware system the input and output hardware are assigned to prescribed processors.

Delays Delays are special operators in the sense that they consist of two parts: a delay input (where new data is put into the delay) and delay output (where new data is retrieved from the delay). Rafael always treats delay parts as two distinct operations. It is guaranteed, however, that output of a delay be scheduled always before the input of the same delay.

Each node input/output can have a type. Type is a character string which is checked for matching when node inputs/outputs are connected. Rafael allows dynamic type names resolved in compile-time that match to every static type name and solves the type name ambiguities. In Rafael dynamic type names start with the 'TYPE' string, for example 'TYPE23' is a dynamic type string. An adder that can add any type of data can have 'TYPE23' type of each input/output node. When any of the inputs/outputs is connected to an output/input with static type, the dynamic type is replaced by the static type by the checker. For example if the output of the hypothetical adder above is connected to an input node with 'TIME' type, 'TYPE23' is replaced by 'TIME' for all the adder inputs/uotputs and type checking continues on the inputs. *Fig. 1* illustrates the process.

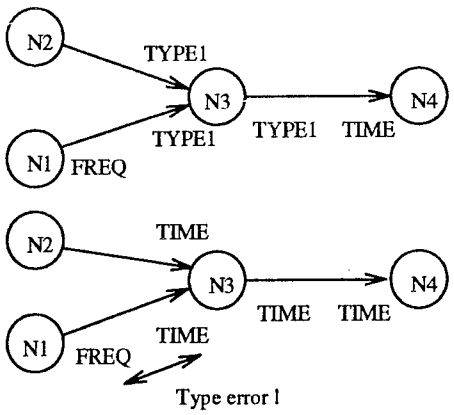


Fig. 3. Propagating type names in Rafael

Depending on the operation library, 'tokens' can have arbitrary size. The actual Rafael operation library supports one-dimensional vector tokens.

6. Rafael Software Model

Rafael accepts a restricted version of synchronous dataflow graphs (LEE – MESSERSCHMITT, 1987) for scheduling. This restriction means that if a node output produces or input consumes more than one token, it can be connected only to an input or output that consumes or produces one token. See *Fig. 4* for example. This simplified scheme allows Rafael to support practically relevant upsampling/downsampling operations without getting to a problematic loop scheduling problem (BHATTACHARYYA – LEE, 1994).

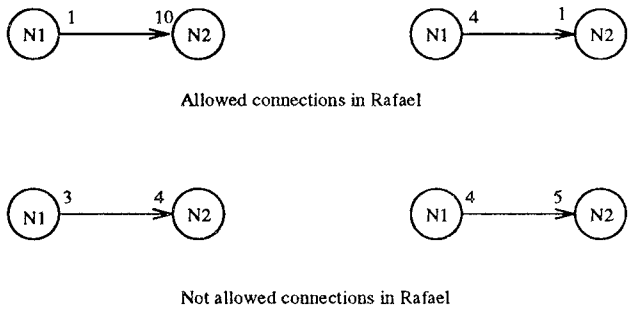


Fig. 4. Rafael's restricted synchronous dataflow graph

Rafael has two software models. The first one is a classical synchronous dataflow model which does not allow run-time decisions. This model has been proved to be too restrictive but this is the most effective one. It allows all kinds of supported operations in the dataflow graph but no conditional structures are permitted, we will call it *static model* in the future. The static scheduler will be invoked for this graph and a single-block schedule will be generated. This model is the restricted version of the second one that allows run-time decisions.

Based on the conditioned dataflow model of synchronous languages a *conditioned block dataflow model* was implemented in Rafael, we will call it *dynamic model.* Inserting `if ... endif` constructs around each operation and considering all conditions *true* it is an evident but not too efficient solution for the run-time decision problem. Instead Rafael forces the SFG designer to *group* parts of the graph to a *block.* A block contains a graph portion for which the following holds true:

1. Inside a block the graph portion is a synchronous dataflow graph without run-time decisions.
2. All the operations in this block depend on the *same condition.*

Outside the blocks only probes and blocks are allowed. This is called *root* level. Operations are embedded into blocks, this is the *block* level.

This simple scheduling scheme used in Rafael solves the scheduling problem in two passes.

1. First it prepares static schedule for each block independently. Variables are propagated through the root level block connections and static scheduler is invoked for the block.
2. Dynamic root-level scheduling. Blocks are considered as operations which run on all the processors at the same time. A list scheduler traverses the block connections and builds the order of the block considering only dependency relations. During the execution a block may or may not be executed depending on its condition input variable (if any).

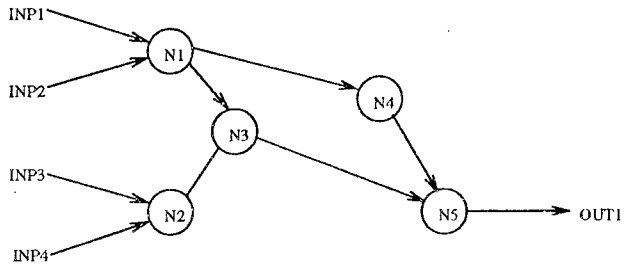


Fig. 5. Example static model graph

Fig. 7 demonstrates this method on the example dynamic model graph in *Fig. 6*.

Advantages of the conditioned block schedule are the following:

- We can provide conditional structures while preserving static scheduling.
- The user of the system is forced to group nodes with the same condition together, the performance loss resulting from the repeated conditional statements is thus avoided.
- The static scheduling algorithm estimates the reality much better than in the SynDEx case. As a block contains only synchronous dataflow, the static scheduling is always exact, not only in the worst case as in SynDEx.
- SIGNAL compiler makes readily the operation grouping itself.

We have to mention the following disadvantages:

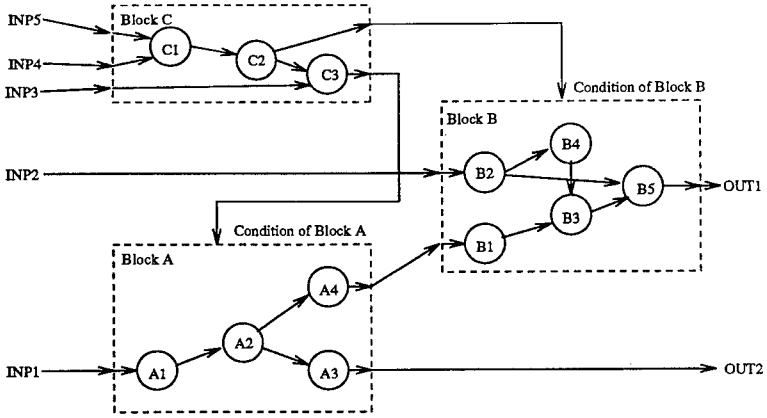


Fig. 6. Example of dynamic model graph

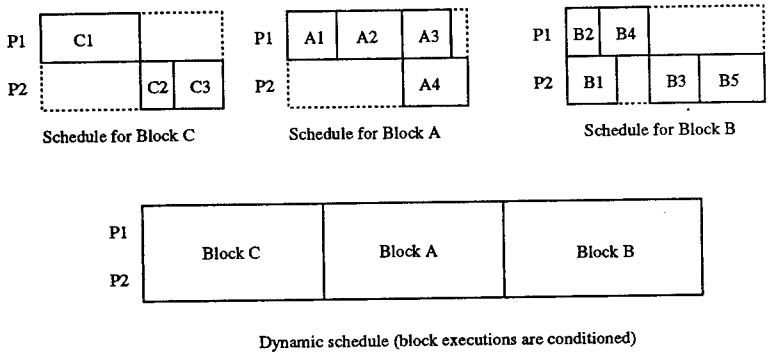


Fig. 7. Example of dynamic model scheduling

- If the blocks contain insufficient operations, static schedules of blocks can be too sparse. In this case even true dynamic scheduling could provide a better solution.
- It is very easy to construct an incorrect graph. Consider the graph in *Fig. 8.* In this example Block B depends on Block A and in the root-level dynamic scheduling it is scheduled after Block A. It cannot be guaranteed, however, that Block A was really executed because it depends on a run-time decision. If the condition of Block A is not true, Block B will get its input from obsolete temporary variables producing a bad result. As Rafael makes no effort to check the cal-

culation of condition variables, these situations cannot be signaled by the compiler.

- Other effect of the fact that Rafael does not analyse the condition calculation is that all the condition variables must be recalculated in each iteration. We can recall that SIGNAL compiler laboriously optimizes the condition tree so that its output program can be the 'laziest' which means that `if ... endif` structures belonging to a clock expression on the lower level of the clock tree will be appropriately nested into `if ... endifs` of upper level clocks. The scheme presented above will flatten the clock tree putting all clock expressions to level 1.

In spite of the disadvantages we consider that the Rafael conditioned block model avoids successfully the dynamic scheduling and in the case of large static blocks and few decisions (which is often true at a DSP algorithm) it is sufficiently efficient.

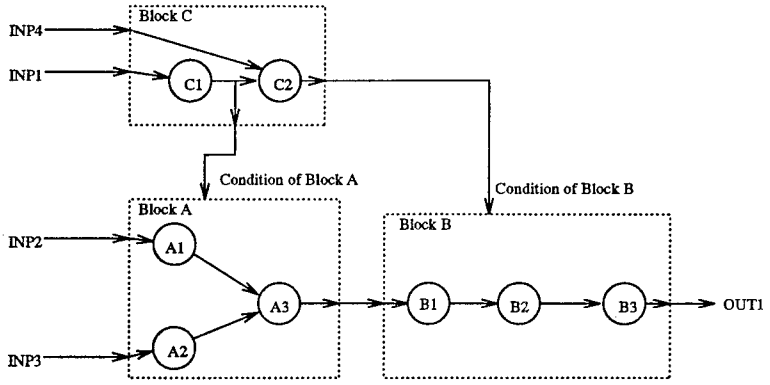


Fig. 8. Example of possibly erroneous graph

7. Rafael Hardware Model

Rafael supposes an arbitrary number of interconnected, heterogeneous processors as target system. The communication hardware connecting these processors can be heterogeneous as well. The static scheduling algorithm prescribes, however, that execution times of operations on all the processors of the target system and communication times on all the channels in the target system should be known in advance. These calculation/communication times can depend on certain parameters, in the case of calculations these

parameters are defined by the operation type, in the case of communication it depends on the amount of data units passed between the processors.

Rafael uses a simplified communication model, critiques say it is oversimplified. Rafael considers the communication structure totally interconnected but allows different communication costs for both directions of each channel. The actual Rafael implementation does not have router algorithm so if the target architecture is not totally interconnected, virtual communication layer must be provided by operation library programmer.

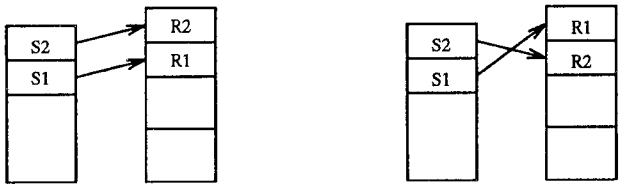
The basic Rafael communication notion is the *channel.* Channels are resources that are shared by processor pairs willing to communicate. A channel is assigned to each processor pair and that channel is occupied for the length of the communication between that processor pair. Other processor pairs having the same channel number have to wait with their request until the channel is free. Channels represent hardware resources used for communication (bus, network, communication links, etc.). The processor pair–channel number assignment is fixed in the hardware database.

Each communication activity can have three properties which are returned by the hardware database functions to the compiler core.

Activity time It is the time during which the communication activity occupies the processor it is scheduled on. If the communication hardware needs constant interaction with the processor (buffered serial line hardware, for example) the activity time is the same as the time required for the communication activity. In the case of DMA it is the DMA initialization time.

Survive time This is the time which is needed to finish the communication after the activity itself finishes. For example a DMA is initialized during the activity time then it accomplishes the task. During the survive time the variable which is sent cannot be reused and no new communication activities can be accomplished on that channel. On the receiving side all the calculations which need the received variable are delayed until the end of the survive time.

Synchronous flag This flag controls the scheduling of communication activities. If this flag is false for a certain communication activity, the scheduler can put the send activity before the receive activity of the same communication pair. No 'crosses' are allowed, however (see *Fig. 9*). If the synchronous flag is true, the send and receive activities are scheduled strictly at the same time.



Valid non-synchronous communication activity arrangement

Invalid arrangement ("cross")

Fig. 9. Allowed and not allowed communication schemes

8. Graph Description Language

The actual Rafael implementation does not contain a graph editor, the user must construct the input algorithm graph himself or herself. A simple graph description language is used for this purpose which will be described briefly in this section.

According to the two software models in Rafael. there are two variations of the graph description language. In the first variation (synchronous dataflow) only probes, nodes, delays and connections are allowed. Let us see an example graph:

```
PROBE I 1 1 A_TYPE 1 1
PROBE I 2 1 A_TYPE 1 1
PROBE 0 7 1
NODE 4 ADD (4)
NODE 5 ADD (4)
NODE 6 ADD (4)
NODE 8 MUL (4)
NODE 3 CONST ((1 2 3 4))
DELAY 9 4 1
CONNECTION 1_1 4_1
CONNECTION 2_1 4_2
CONNECTION 2_1 5_1
CONNECTION 3_1 5_2
CONNECTION 4_1 6_1
CONNECTION 5_1 6_2
CONNECTION 6_1 8_1
CONNECTION 3_1 9_1
CONNECTION 9_1 8_2
CONNECTION 8_1 7_1
```

PROBE <I/O> <nodenum> <type> <upsample> <downsample>
<I/O> is the input/output probe type, <nodenum> is the number of the node, <type> is its type name. For convenience of the compiler, Rafael stores the relative sample rate of the node in rational form. <upsample> is the nominator, <downsample> is the denominator of the relative sample rate (see section 11).

NODE <nodenum> <operation> <parameters>
<nodenum> is the node number, <operation> is the function attached to the node, <parameters> is the parameter list which depends on the function. In the case of the example ADD operator determines the size of the vectors to be added.

DELAY <nodenum> <delay size> <delay length>
<nodenum> is the number of the node, <delay size> is the size of one token it stores, <delay legth> is the number of delay stages data fed into the delay goes through. Delays explicitly have TYPE inputs/output types.

CONNECTION <onode>_<onum> <inode>_<inum>
Defines a connection between the output numbered <onum> of the node having <onode> node number and an input described by similar parameters.

The conditioned block dataflow model allows block definitions beside the elements above. In this model only probes, block definitions and connection definitions are permitted at root level.

```
BLOCK MADD2  I1->6_1:TYPE1 I2->5_2:  TYPE1
             I3->5_1:TYPE1 01->6_1:  TYPE1
NODE 5 MUL (4)
NODE 6 ADD (4)
CONNECTION 5_1 6_2
ENDBLOCK MADD2


BLOCK MUL2  C:BOOL I1->6_1:TYPE1 I2->5_2:  TYPE1
            I3->5_1:TYPE1 01->6_1:  TYPE1
NODE 5 MUL (4)
NODE 6 MUL (4)
CONNECTION 5_1 6_2
ENDBLOCK MUL2


PROBE I 1 1 A_TYPE 1 1
PROBE I 2 1 A_TYPE 1 1
```

```
PROBE I 3 1 A_TYPE 1 1
PROBE I 10 1 BOOL 1 1
PROBE 0 7 1

NODE 4 MADD2
NODE 5 MUL2
CONNECTION 10_1 5_C
CONNECTION 1_1 4_1
CONNECTION 2_1 4_2
CONNECTION 3_1 4_3
CONNECTION 1_1 5_1
CONNECTION 2_1 5_2
CONNECTION 4_1 5_3
CONNECTION 5_1 7_1
```

The only new element is the BLOCK ... ENDBLOCK definition pair. Blocks group their internal nodes into one virtual operator that can be placed by a NODE definition. A internal node in a block is identified by its block name and node number, two blocks can have internal nodes with the same node number as internal nodes are invisible outside of a block. The block header contains the following elements:

I <inputnum> - ><inp nodenum>_<inp inputnum>:<typename>
Connects <inputnum> input of the virtual operator represented by the block to <inp inputnum> input of <inp nodenum> internal node. Type of the block's input is set to <typename>. Data fed into that input of the block will be propagated to the internal node's input.

O <onum> - ><onodenum>_<out outputnum>:<typename>
Connects <onum> output of the virtual operator represented by the block to <out outputnum> output of <onodenum> internal node. Type of the block's output is set to <typename>. Data produced by that output of the internal node will be propagated through the output of the virtual node.

C :<typename> Indicates that the block has condition input and the type of the condition input is <typename>. Condition input can be referenced as 'C' in the CONNECTION definition.

9. The Database

Rafael provides a programmable operation and hardware database stored in Lisp. The database is accessed by the compiler core through Lisp functions. The interface of these Lisp functions is documented so that the database programmer can interface to the compiler core.

The database consists of two parts: operation database and hardware database. Operation database stores the actual function set for all the supported hardware devices while hardware database provides Lisp functions that can calculate every characteristic of the target hardware system which is necessary for scheduling and code generation.

The database is handled and maintained through the XLisp interpreter and stored in Lisp lists. Because XLisp runs under Windows, all its memory is virtualized so we can store the whole database in the memory of XLisp. The simplifies greatly the implementation of the database management because we simply use the built-in list manipulating functions of LISP.

The Operation Database

The operation database has two parts: operator headers and compilation strategy functions. The operator headers are stored in lists which are bound to the operator name. This list stores the following information:

- The name of the compilation strategy routine.
- The description of the input(s) (type, size).
- The description of the output(s) (type, size, storage class, sample rate factor).
- The execution time in system clock beats.
- Parameters. The parameters and their meaning are defined by the creator of the operator library. For example the parameters for the FIR operator can be the length of the filter and the filter coefficients. The actual values of the parameters are supplied when the user places an operator, it is passed in the SFG script.
- Constructor and destructor routines. The compiler creates a constructor function for each operator which requests it. The constructors are invoked before the operator is executed first time. Similarly, before the SFG execution terminates, destructor functions are called for the operators which need it.

  The data structure above is described in a list like the following:

```
(  ( strategy list)
   ( inputs )
   ( outputs )
   ( time function )
   ( parameters ) )
   ( constructor strategy list )
   ( destructor strategy list )
)
```

The strategy list contains the names of the compilation strategy functions for each hardware device. It has the following format:

```
(  (device1 function1) (device2 function2)
   ... (deviceN functionN) )
```

The compilation strategy function is called each time during the code generation pass when the schedule contains a reference to that function and its program text must be generated. This LISP function gets the label lists of the input and output branch descriptors (effectively labels of data areas where the compiler allocated space for the temporary variables), the parameter list (which contains data like coefficient vector of a filter, etc.) and returns the program text to the compiler which writes it into the output file. The strategy function can decide on the subroutine chosen or the form of the generated program text depending on the input and output connections and the actual parameters. The subroutine bodies can be stored in an ordinary object library, in this case Rafael will place only references into the code which can be resolved by the linker which belongs to the DSP's development system. This subroutine library can be created and maintained by the assembler and library manager tools of the DSP development software package. Another design style is to inline all the operation bodies which result in slightly faster code but larger code size.

The excellent symbol handling capability of the LISP which makes this language so appropriate for the artifical intelligence applications can be exploited in this system and we can build significant intelligence into the strategy functions.

The input list stores the description of the operator's input. Its format is the following:

```
(  ( type1 size1 ) (type2 size2) ...
   (typeN sizeN) )
```

where type is the freely chosen signal type (for example time for time domain signals) and size is the size of the input vector accepted by this node. This size can also be a symbol from the parameter list (for example the size of an FFT input can be `N` where `N` is a parameter supplied by the SFG designer) or even a lambda function of the parameters. The type name can be either static or dynamic. Dynamic type names have the form of 'TYPEn' where n is an integer number. Dynamic type names are resolved when they are connected to a statical one.

The output list is similar, but beside type and size it also contains the storage class specifier and the upsample and downsample factors. Its format is the following:

```
(  ( type1 size1 st1 us1 ds1)
   (type2 size2 st2 us2 ds2) ...
   (typeN sizeN stN usN dsN) )
```

The storage class specifier shows whether the compiler has to allocate space for the output variable or the space is reserved by the operator. The us and ds values describe the change in sampling frequency caused by the operator. The us denotes the multiplication, ds is the division of the sampling frequency. For example the pair 2 1 means interpolation by 2.

The time function list stores Lisp functions which get the bound parameter list and return the execution time of the operator on a given hardware. The list has the following format:

```
(  ( device1 lambda1 ) ( device2 lambda2 ) ...
   ( deviceN lambdaN ) )
```

where `lambda1 ... lambdaN` are lambda expressions (no-header Lisp functions) which compute the execution time for the given device.

The parameter list contains operator-dependent data. For example in the case of an IIR filter it contains the size of the nominator and denominator coefficient vectors and the vectors themselves. In the operator header the list is stored in unbound form (without parameter values), the editor evaluates this list when placing an operator. The IIR parameter list would look like the following in unbound form:

```
(N COEF1 M COEF2) )
```

and in bound form (after the operator has been placed)

```
(3 (0.34 - 0.2 2.12) 4 (0.23 0.77 0.192 2.94) )
```

This bound form is stored in the SFG description file and is passed to the execution time computing and strategy functions when necessary.

The constructor and destructor strategy lists have the same format as the strategy function. An operator may have constructor and/or destructor functions — pieces of code which are executed before the operator's first run and after the operator's last run. If the operator does not need such functions, NIL is stored instead of the name.

The following small code piece shows the implementation of the ADD database entry for the TMS320C30 and DSP96002.

```
(setq add ' ((
; c30add is C30 strategy function
                  (c30 c30add)
; dsp96kadd is 96K strategy function
         (dsp96k dsp96kadd)
         )
; Has two inputs, each of size n
; (n is the operation parameter)
                  ((type1 n) (type1 n) )
; Has one output, size n, automatic storage,
; interpolating factor:  1
                  ( (type1 n a 1 1) )
; Time functions for C30 ...
                  ( (c30 (+ (* 2 n) 10) )
; and 96K
                  (dsp96k (+ (* 2 n) 5) )
                  )
; Has only one parameter (n)
                  ( n )
; No constructor for C30 and 96K
                  ( (c30 nil) (dsp96k nil) )
; No destructor for C30 and 96K
                  ( (c30 nil) (dsp96k nil) )
         )
)
```

Target Hardware Database

The target hardware database provides the following information to the compiler core:

- Processor numbers and processor types in the target system.
- Activity, survive times and synchronization flag for any communication activity.
- Communication cost estimation for any communication path in the target system (for the scheduler).
- Channel-processor pair assignment for any processor pair.

A set of Lisp functions must be written for each target system. It is a relatively inconvenient solution but allows greater flexibility.

10. Rafael Memory Management

Rafael allocates memory for temporary variables in compile time. When the generated program runs on the target system, every variable is already assigned a memory address. Rafael implements a simple 'first fit' dynamic memory allocation scheme when compiling the graph.

When a node is scheduled, Rafael allocates its output variables (the input variables must have already been allocated). The scheduler keeps track of the actual state of memory map by the means of chunk lists which describe, actually what size of blocks are occupied at what address in the memory of the target processor. When allocating a variable the memory manager simply walks this chain and finds the memory block with the lowest address which is big enough to accommodate the variable to be allocated.

When an output variable is created, its 'scope' is established. A variable goes out of scope if all the operations that consume this variable hase already been executed. In this case the memory chunk assigned to the variable is freed and the place the variable occupied can be reused. As the scheduler cannot know when allocating the variable, on which processor(s) that variable will be consumed. every instance (variable sent to other processors) of that variable stays 'alive' on every processor until all operations that consume that variable terminate.

A variable can be local or global. Local variables are used internally by blocks. A variable is local if it is created in a block not at root level and it is consumed only by the operations of that block (so it is not connected to a block output). Every other variable is global. Blocks have their own address maps that start at relative address 0. At the end of the scheduling when we know, how much memory is required fir the global variables, local variable addresses are relocated so that these variables be allocated starting at the end of the memory allocated for global variables. Local variables of blocks thus overlay each other (*Fig. 10*).

11. Compiler Passes

Rafael compiler works in 5 passes.

Reading Graph Description File

The compiler reads in the SFG file and parses it syntactically. Then it analyses the connection definitions and signals connection errors (connecting to

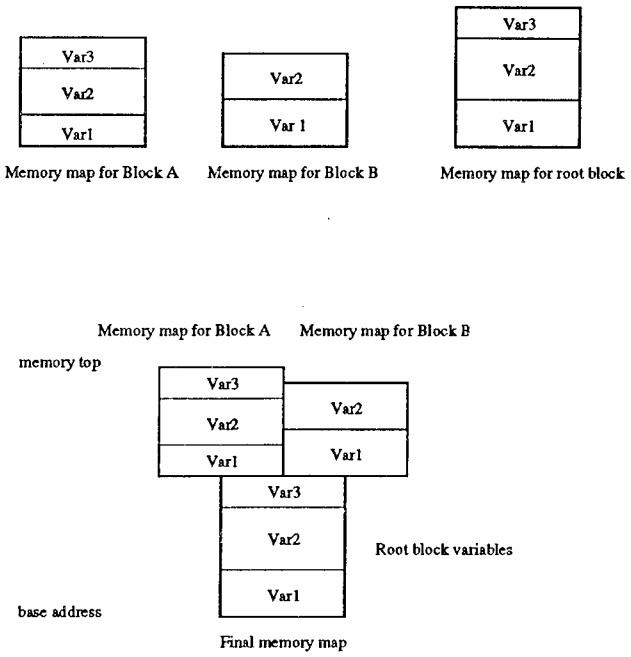


Fig. 10. Block memory overlaying in Rafael (supposing 1 processor)

nonexisting node, nonexisting input, etc.). During this phase the compiler rebuilds the tree in the memory of the computer, ready for analysis.

Type Checking

The compiler resolves the dynamic type names and checks if there are type errors (see section 4 for further explanation). The type checker is a recursive routine that propagates the static type names from node to node substituting dynamic type names with static ones and signaling errors if type name violation is found. The type checking starts at descendants of probes as they are the only nodes that surely do not have dynamic types.

IPF Checking

IPF stands for interpolation factor and is used to support Rafael's multirate features (section 6). IPF is the rate of the node's execution in the multirate

model. IPF is represented by two distinct numbers, the nominator and the denominator so IPF:1.4 means 1/4 execution rate.

Rafael uses a recursive subroutine similar to the typechecker to propagate IPFs along the graph and looks for the minimal IPF factor. Propagating IPF means that the IPF at the input of the operation is multiplied by the sample frequency multiplication factor stored in the database at the output description yielding output IPF then it is passed to all the nodes connected to the outputs. The actual implementation of Rafael prescribes that the output sample on all the outputs should be the same. During the IPF propagation the minimal IPF in the graph is recorded. As IPF is calculated by division or multiplication by integer factor, all IPFs in the graph must be integer multiple of the minimal IPF. So the factor

$$c_{loop} = \frac{IPF_{node}}{IPF_{min}}$$

is the loop count that determines, how many times an operation with IPF IPF_{node} must be repeated if the minimal IPF is IPF_{min}. Note that operation changing IPF are always executed on the higher input sample rate and output sample rates (*Fig. 11*).

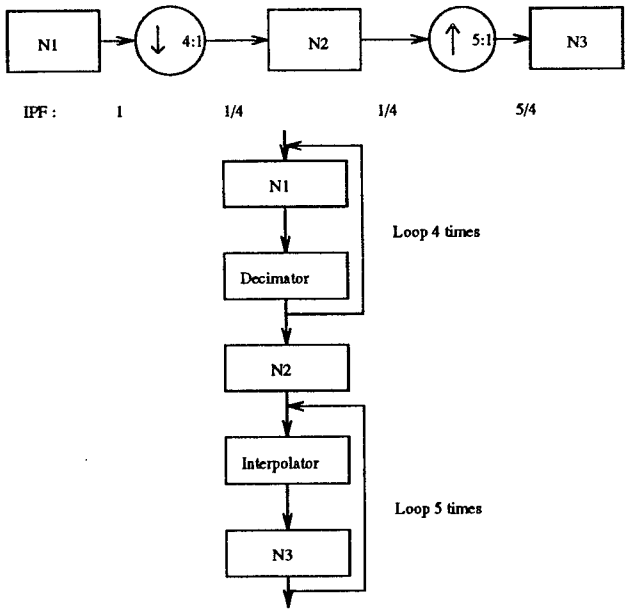


Fig. 11. IPFs in an example graph and looped schedule

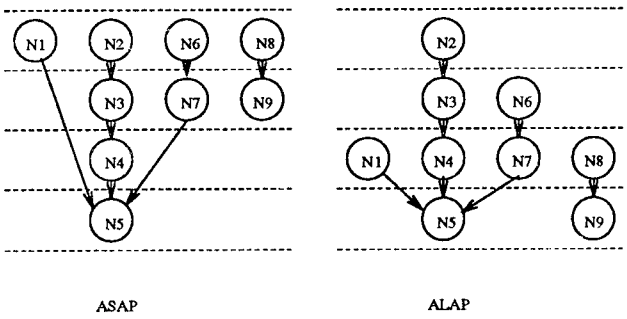


Fig. 12. ASAP and ALAP schedules

Scheduling

The formally correct, typechecked graph with IPF values for all the nodes calculated is then passed to the scheduler algorithm. The actual version of Rafael contains only the RHLS scheduler but work is under way to implement the much more efficient Springplay scheduler (PALLER – WOLINSKI, 1995) in the software.

RHLS is an ALAP-based list scheduler which was made suitable for heterogeneous environment. In the first step we create ASAP and ALAP schedules in order to get the ALAP levels. We present briefly ASAP and ALAP schedules below.

ASAP algorithm was presented first in HU's classical publication (HU, 1961). ASAP scheduler starts operations as soon as all the predecessor nodes terminate the computation that is

$$E(n_i) = \max(E(\text{pred}(n_i))) + t_{n_i}^{e,asap}, \tag{1}$$

where $t_{n_i}^{e,asap}$ is the execution time of node i and $E(n_i)$ is the earliest time when n_i can be executed. Node with no predecessors have $E = 0$. This simple version is only for homogeneous architectures. The original version supposes unlimited resources and schedules nodes just at their E.

ALAP schedule is based on very similar principles. Nodes are scheduled as late as possible without increasing the length of the schedule.

$$L(n_i) = \min(L(\text{succ}(n_i))) - t_{n_i}^{e,asap}. \tag{2}$$

$L(n_i)$ is the latest time when n_i can be executed in the case of minimal length schedule. L values of nodes with no successors are initialized to the maximal E value over the entire graph. *Fig. 12* depicts the ASAP and ALAP schedules of an example graph.

RHLS assume the we can always schedule the nodes on the fastest processor possible so minimum execution time is supposed when building the ASAP-ALAP schedules.

$$t_{n_i}^{e,asap} = \min(\bar{t}_n^e)\,,$$

where t_n^e is the execution time vector that is composed of execution time of node n on each processor. Then we define *urgency* of the operation n like the following:

$$u_{n_i} = L_{n_i} - t_v\,, \tag{3}$$

where t_v is the *virtual time* and it will be detailed later.

The base of the scheduling heuristic is to assign the nodes on the critical path to the fastest processor available. The more urgent it is to execute a node (as its delaying would set back the execution of the whole graph) the faster processor it deserves. The most urgent nodes are those which have the lowest ALAP time.

We pick hence the node to be scheduled based on the u_{n_i} urgency value defined above (lowest urgency value means more urgent node) and we need the best processor to execute it. The best processor selection is very simple: we try the node on each processor considering the communication costs and we pick the one on which the node achieves the earliest completion time. Before trying a node on a processor, necessary communication activities are scheduled tentatively so that we know how much time must be calculated for fetching the input variables produced on other processors.

The heuristic algorithm works like the following:

```
Create the ready node list from nodes that have no predecessors;
while the ready list is not empty do
     for all nodes do
          if u(i) < minimum so far
            Candidate = node i;
     end for
     Try the candidate on each processor considering communication
          cost;
     Choose the processor on which the task achieves the earliest
          ending time;
     Schedule candidate node and the necessary communication
          activities on candidate processor;
     Update u(i)s and tv;
     Add nodes that become ready to the ready list;
end while
```

As the real t_{n_i} node starting times will generally not be equal to the ideal ASAP or ALAP starting times the scheduler maintains *real processor*

times and t_v *virtual time.* The virtual time is used to track the time in the ALAP schedule graph while the real time is the scheduling time on the processors. The t_v variable shows where we are in the ALAP schedule graph, it is set to the lowest ALAP time among the ready nodes. The last step is the updating of urgency and virtual time variables.

The version implemented in Rafael differs from the algorithm presented above considers node repetition resulted by multiple sample rate loops (see IPF checking section). The schedulers consider effective node execution time as $c_{loop} \cdot t_n^{e,asap}$ and try to group nodes with the same IPF together.

Code Generation

The scheduling done, Rafael generates the output text for each processor. The code generator walks the activity list on each processor then asks the Lisp code generator database functions to produce output text for them which is then sent to the output file. Separate output files are generated for each processor. The model of output text will be discussed in detail in the next section.

Code Generation Model

Rafael has a parametrizable code generation that allows each section of the text generated to be redefined. The code generator invokes Lisp functions that receive the parameters of the text section and the device for which the code will be generated then it is the responsibility of these Lisp functions to produce the appropriate text. These code pieces are called *code generator service functions* and they complement the operation strategy routines. Every text section that Rafael writes to the output text file can be redefined by modifying either the operation strategy functions (in the case of operation texts) or the code generation service functions (headers, communication routine codes, etc.).

Rafael generates three text sections for each processor (that may be empty as well). For programmable processor-like devices that Rafael was designed for, the database programmer may wish to realize these three sections as subroutines. These sections are the following:

1. Constructor section. Called only once from the user program before the first iteration of the dataflow computation.

2. Operation section. Called once for each iteration. Calling the operation section entry label will actually execute the program generated from the SFG.
3. Destructor section. Called once after the last iteration of the operation section.

Each section has a start and end header that probably contain section head label in the start header and 'return' instruction in the end header. The sections contain the text generated by the operation constructor, strategy and destructor functions.

If the compiled SFG was written in block conditioned model, each section has a separated part for each block. In the constructor and destructor sections it is rather a formality as Rafael guarantees no specific order among the operators when it generates constructor and destructor sections. In the operation section each block has a start and end header. The current operation library realizes blocks as subroutines so the start header defines a block entry point label and the end header contains a 'return' statement. The block subroutine contains the operation body texts in the schedule order. Having block subroutines generated, Rafael emits the text for the root block that contains probe calls and block invocations. Block invocations in the current operation library result in subroutine 'calls' to block subroutines.

12. Conclusions

Rafael cannot compete in complexity with the most advanced systems partly because of the limited capabilities of the host computer we chose, partly because of the significantly less human resources we could devote to the project. The final product, the compiler itself has been implemented but many support programs that would make its usage convenient have not even been planned. For this reason the actual Rafael system is not so 'user-friendly'. As all the resources were concentrated on the compiler development, important parts of the system have not achieved the necessary level yet. The most important among them is the operation database that contains only about a dozen operations only for the TMS320C30 and DSP96002 DSPs. A brave user of Rafael must face the immediate task of filling up the database which requires Lisp programming. Lisp is considered a difficult language among the users although the simple functions needed by the compiler core should be easy to implement for a bit more experienced programmer.

Two distinct influences can be discovered in the Rafael design. The first one is Lee's synchronous dataflow approach and the Gabriel system

which gave us the first notions, how Rafael should look like. We quickly faced, however, the need of run-time decisions and the difficulties it causes in a system based on synchronous dataflow. The second influence that we embedded into Rafael was the way the synchronous language compilers work and SynDEx transformes their output to distributed code. Critique of the SynDEx approach was given and a model that was easy to implement to an existing synchronous dataflow system was developed and realized. Limits of this model were pointed out but we consider that in many practical cases, notably in the DSP case they are acceptable. Further researches are conducted to find a better way for handling dynamic structures in a dataflow system.

So Rafael project achieved its aims at the following points:

- A flexible multi-target SDF compiler has been realized on PC platform.
- Effective scheduling algorithms have been developed for the heterogeneous case.

Rafael still has a long way to go at the following fields:

- More user-friendly environment (graph editor, database editor tools, etc.).
- Complete database for various DSP processors.
- Better communication model.

References

AMAGBEGNON, T. – BESNARD, L. – LE GUERNIC, P.: Arborescent Canonical Form of Boolean Expressions, *INRIA Research Report.* No. 2290. June, 1994.

BHATTACHARYYA, S. S. – LEE, E. A.: Memory Management for Dataflow Programming of Multirate Signal Processing Algorithms, *IEEE Transactions on Signal Processing,* Vol. 42, No. 5, pp. 1190–1201, May 1994.

BURNAI, P. – LAVERENNE, C. – LE GUERNIC, P. – MAFFEÏS, O. – SOREL, Y. (1994): Interface SIGNAL-SynDEx, INRIA RESEARCH REPORT,No. 2206.

BOUSSINOT, F. – SIMONE, R. (1991): The ESTEREL Language, *Priceedings of IEEE,* Vol. 79, No. 9. pp. 1293–1303.

BUCK, J. T. – HA, S. – LEE, E. A. – MESSERSCHMITT, D. (1991): Multirate Signal Processing in Ptolemy, *Proc. IEEE ICASSP-91* Toronto, Canada, April 1991.

BUCK, J. T. (1993): Scheduling Dynamic Dataflow Graphs with Bounded Memory Using the Token FDlow Model, Ph.D. dissertation, University of California at Berkeley.

BUCK, J. T. – HA, S. – LEE, E. A. – MESSERSCHMITT, D. G. (1994). A Framework for Simulating and Prototyping Heterogeneous Systems, *International Journal of Computer Simulation,* special issue on Simulation Software Development, January, 1994.

GENIN, D. – HILFINGER, P. – RABAEY, J. – SCHEERS, C. – DE MAN, H. (1990). DSP Specification Using the Silage Language, *ICASSP-90,* pp. 1057–1060, Albuquwrque, April, 1990.

HALBWACHS, N. – CASPI, P. – RAYMOND, P. – PILAUD, D. (1991): The Synchronous Data Flow Programming Language LUSTRE, *Proceedings of IEEE,* Vol. 79, No. 9. pp. 1305–1319, September 1991.

LANNEAR, D. (1993): Design Models and Data-Path Mapping for Signal Processing Architectures, Ph.D. dissertation, Katholike Universiteit Leuven, March 1993. LEARY, K. W. – WADDINGTON, W. (1990): DSP/C: A Standard High Level Language for DSP and Numeric processing, *IEEE ICASSP-90,* pp. 1065–1068, Albuquerqu, New Mexico, April 1990.

LEE, E. A. – MESSERSCHMITT, D. G. (1987): Szazic scheduling of Synchronous Data Flow Programs for Digital Signal Processing, *IEEE Transactions on Computers,* pp. 25–35, Vol. C-36, No. 1, January 1987.

LEE, E. A. – HO, W-H. – GOEI, E. E. – BIER, J. C. – BHATTACHARRYYA, S. (1989): Gabriel: A Design Environment for DSP, *IEEE Trans. on Acoustics, Speech and Signal Processing,* Vol. 37, No. 11, November 1989.

LE GUERNIC, P. – GAUTIER, T. – LE BORGNE, M. – LE MAIRE, C. (1991): Programming Real-Time Applications with SIGNAL, *Proceedings of IEEE,* Vol. 79, No. 9, pp. 1321–1336, September 1991.

HU, T. C. (1961): Parallel Sequencing and Assembly Line problems, *Oper. Res.,* Vol. 9, pp. 841–848, November 1961.

DE MAN, H. – RABAEY, J. – SIX, P. – CLAESEN, L. (1986): Cathedral-II, A Silicon Compiler for Digital Signal Processing, *IEEE Design @ test,* pp. 13–24, December 1986.

MESSERSCHMITT, D. G. (1984): A tool for structured functional simulation, *IEEE J. Selected Areas of Communication,* Vol. SAC- 2, Jan. 1984.

PALLER, G. – WOLINSKI, C. (1995): A New Class of Compile-Time Scheduling Algorithm for Heterogeneous Target Architectures, *IFAC/IFIP Workshopon Real Time Programming,* Fort Lauderdale, November 1995.

SOREL, Y. (1994): Massively Parallel Computing Systems with Real Time Constraints – The Algorithm Architecture Adequation Methodology, *Proc. Massively Parallel Computing Systems,* Ischia, May 1994.

PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 41, NO. 3, PP. 231-239 (1997)

AN EXPERIMENTAL INVESTIGATION OF A MULTI-PROCESSOR SCHEDULING SYSTEM

Colin REEVES and Helen KARATZA*

School of Mathematical and Information Sciences
Coventry University, UK
Email: CRReeves@uk.ac.cov.cck
* Department of Mathematics
Aristotle University of Thessaloniki
Thessaloniki, Greece
Email: cbdz05@grtheunl1.earn

Received: June 23, 1995

Abstract

The scheduling of jobs through a multi-processor system is important from many aspects. It is often assumed that jobs are scheduled on the basis of some simple rule, such as First-Come First-Served, or Shortest Processing Time First.

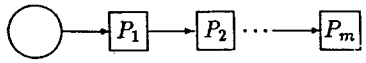
In earlier work we found some evidence to suggest that use of a more sophisticated strategy, based on the use of a Genetic Algorithm (GA) to 'look ahead', could enhance system performance. Here we investigate this idea more thoroughly.

1. Introduction

Recent advanced in heuristic methods for static sequencing problems have included several reports [1, 2, 3] of the use of *genetic algorithms,* which have been found to be robust and efficient ways of solving such problems. In an earlier paper [4], we considered the use of a genetic algorithm (GA) to solve a dynamic flowshop sequencing problem. This problem relates to the sequencing of jobs on machines in a manufacturing environment, but this case has obvious parallels in a computing context, where the jobs are program tasks, and the machines are processors. There is a difference in that the scheduling of computer program tasks needs to be done in real time, which is not so critical a requirement in a manufacturing environment. But first we need to establish whether such an approach can indeed out-perform simple scheduling rules, before considering how it could be implemented in practice.

We consider a somewhat idealised problem, where jobs arrive at a job pool before passing through m processors arranged in series. The time t_{ij} required for processing job i on processor j is known, or can be reliably estimated. There is infinite buffer storage between consecutive processors, and no job pre-emption is allowed. Initially there are n' jobs in the pool,

but further jobs arrive as time passes, in accordance with a known inter-arrival time probability distribution.



The problem is at any stage to determine the sequence in which the jobs in the pool should be processed in order to optimise some measure of performance. This is clearly a dynamic problem, since as more jobs arrive the current 'best' sequence may have to change. Of course this is an approximation to what really happens — in real problems jobs may not call on all processors in the same order, they may need to visit a subset of processors more than once, and so on. However, our purpose in studying this simplified version of the problem, as outlined above, was to test the effectiveness of different ways of scheduling jobs. Simple scheduling rules are usually concerned only with the next job, without trying to consider the current job pool as a whole. Our hypothesis is that using a GA to 'look ahead' would be a more effective means of approaching such a problem.

There are a number of ways of assessing the performance of a system like that described. It was decided that the most natural performance measures would be the mean *response time,*

$$R(n) = \sum_{j=1}^{n} \frac{C_j - A_j}{n},$$

where n jobs have been processed, and job j arrived at time A_j, and was completed at time C_j; and the *throughput rate,*

$$T(n) = \frac{n}{C_n - A_1}.$$

These performance measures are of course correlated to some extent, but while response time refers to the system performance from the viewpoint of the jobs, throughput rate measures performance from the server's perspective.

2. Implementation

A simulation model of the system described above was programmed, as shown in the box below:

- Initialise job pool;
- Compute job sequence;
- Schedule 1st job;
- Compute 1st event time T_E;
- Repeat
  If no arrivals before T_E then
    1. schedule next job;
    2. compute next event time T_E;
- else
    1. add additional job(s) to current job pool;
    2. re-compute job sequence for current pool;
    3. schedule next job
    4. compute next event time T_E;
- Until simulation time exceeds a specified limit.

Clearly, the GA enters at the points where a re-computation of the 'best' sequence of the current job pool is required. The simple scheduling rules would also be implemented at this stage.

It is important to realize that by re-computing the best sequence from the current job pool, we make the assumption that a good overall solution will be obtained if we try at any stage to sequence the currently available jobs as if no more jobs will arrive. It is this hypothesis that we shall evaluate by comparing with more traditional job scheduling criteria.

2.1. The Genetic Algorithm

The GA used to solve the sequencing problem was adapted from that described in [3], whose characteristics can be summarised as follows:

- an initial population of 30 chromosomes using a sequence representation;
- parent selection using ranking;
- incremental population replacement (also known as a steady-state GA);
- replacement of a randomly chosen string of below-median fitness;
  - a sequence-based crossover (sec [3] for details);
  - an adaptive mutation rate;
  - a termination condition of $\mathcal{O}(nm \log[m+n])$

objective function evaluations.

At the first stage, the initial chromosomes were chosen at random, and this could also be done at each subsequent application of the GA. However, by basing, at each stage, the initial population on the population of solutions obtained at the previous stage, we found that good solutions to the current problem were determined more rapidly, which may be an advantage when a decision on the next job to be sequenced is needed in real-time.

2.2. Other Selection Criteria

There were 3 obvious candidates for simple selection criteria instead of the GA: we could use

- job arrival order (FCFS);
- shortest (first machine) processing-time order (SPT(1)).
- shortest (total) processing-time order (SPT(all));

The first of these corresponds to doing nothing, simply scheduling on a First-Come First-Served basis; the other two attempt to take into account the likely delay to other jobs that could be incurred by scheduling a specified job now. Clearly, by scheduling a job with a large processing-time requirement when other (shorter) jobs are available, the response-time for those other jobs is likely to be increased. The first machine is of course the most important in our model, since once the current job completes processing on the first machine, we are free to schedule another. The rationale for SPT (all) is that, like the GA, it also tries to 'look ahead' beyond the immediate decision.

These type of criteria have been studied for some special cases of single-processor scheduling problems using a queuing-theoretic framework, and CONWAY et al. [5] have an interesting discussion which shows that,

under certain conditions, the Shortest-Processing Time criterion is optimal for single-machine problems. However, this cannot be shown to hold for multi-processor problems.

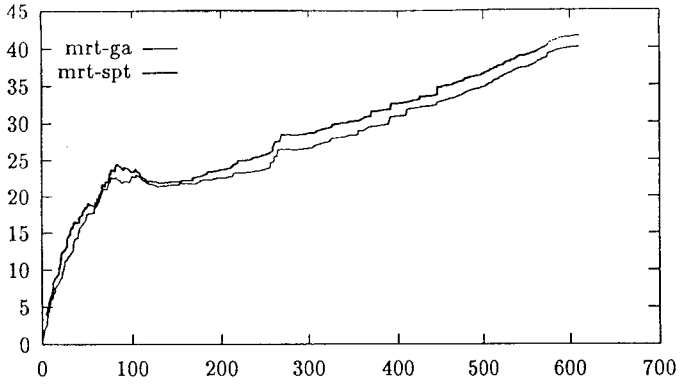


Fig. 1. Mean response times

3. Test Problems

Several sets of test problems were generated. In each case, the arrival rate and service (processing) rate of jobs were assumed to be the same: clearly, if the arrival rate is greater than the service rate, the size of the job pool will increase without bound, which would not be tolerated in a real system. Job arrivals were assumed to occur according to a Poisson process, but job-processing times were generated from 5 different distributions with different coefficients of variation (CVs) of processing-times. We used Erlang-k (hypo-exponential) distributions with $k = 4$ and $k = 16$, an exponential distribution (corresponding to a Poisson process), and two branching-Erlang distributions to simulate distributions with high CVs. The complete range of CV values was {0.25, 0.5, 1, 2, 4}. General details of the distributions used and their characteristics can be found in, for example, SAUER – CHANDY [6].

In each case, 30 jobs were assumed to be in the pool initially, and the simulation was continued until a further 530 jobs had entered the system. In the first group of problems, the number of processors was set at 4. The values of $T(n)$ and $R(n)$ were measured when each job completed all its tasks. They could then be plotted on a graph as shown in the example below.

To do this for every run is clearly impracticable; in order to summarise these graphs, we calculated the average difference between the $T(n)$ and $R(n)$ values for GA and FCFS over the whole run length. Thus we could obtain a measure of the success of the GA against the 'donothing' option. We then repeated this for SPT(1) against FCFS, and SPT(all) against FCFS. Each case was replicated 4 times, so the results reported in *Table 1* below are the means of 4 runs in each case. In this table, the first value in each cell is the average difference for $R(n)$, the second for $T(n)$.

Table 1
Average differences in MRT & TPR: 4 processors

CV	GA	SPT(1)	SPT(all)
0.25	−10.33	−9.15	−12.76
	0.024	0.014	0.034
0.50	−10.48	−8.73	−16.03
	0.023	0.012	0.058
1.00	−17.35	−13.26	−20.37
	0.056	0.032	0.112
2.00	−32.32	−21.05	−39.90
	0.119	0.052	0.201
4.00	−30.76	−6.97	−37.61
	0.141	0.026	0.174

The whole procedure was then repeated for the case of 8 processors, with the results shown in *Table 2.*

Table 2
Average differences in MRT & TPR: 8 processors

CV	GA	SPT(1)	SPT(all)
0.25	−12.52	−8.64	−12.54
	0.038	0.016	0.036
0.50	−8.72	−6.94	−13.03
	0.022	0.007	0.041
1.00	−17.54	−10.06	−21.05
	0.056	0.006	0.097
2.00	−34.58	−12.50	−37.74
	0.090	0.024	0.140
4.00	−62.32	−21.16	−59.71
	0.133	0.028	0.170

On the whole, all 3 rules were able to improve system performance, in terms of both performance metrics: that is, it is possible to improve performance both from the point of view of the system throughput and at the same time to provide a better service from the point of view of the customer.

It is clear that SPT(1) provides the last improvement in performance over FCFS. It can also be seen that SPT(all) nearly always does slightly better than using the GA. This contradicts our earlier findings, reported in [4]. We have not yet fully resolved this contradiction, which may be simply an artefact of the random number streams used in the simulation. However, this latest work is based on more extensive and comprehensive testing, and is probably more reliable. The differences are in any case fairly small.

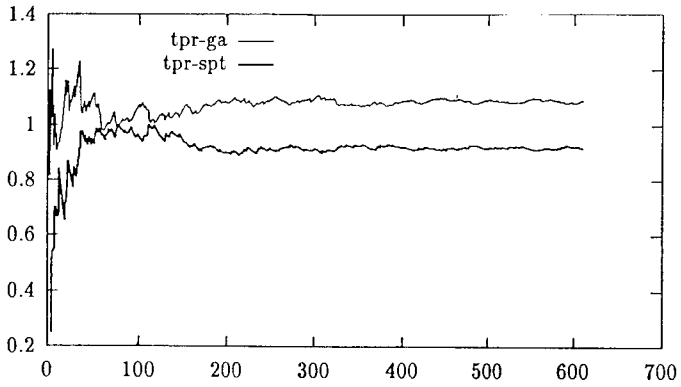


Fig. 2. Throughput rates

We must also bear in mind the amount of computing has to be done in each case. In terms of time actually spent in selecting the next job, SPT(1) needs the least, while SPT(all) needs slightly more, since it requires the summation of processing times for m machines. The amount needed for the GA can be user-controlled, depending on what degree of convergence to the (unknown) optimal sequence is desired. In practice, and on average, it took an order of magnitude more computation than the SPT rules. (There was considerable variability, too: in the case $CV < 1$, for most of the simulation period the queue lengths tended to be much greater, which meant the GA had far bigger problems to solve, and thus took much longer.) In view of this, it would seem desirable to use the simpler SPT(all) rule.

4. Conclusions

The results obtained confirm that, on average, the performance of a multi-processor system is improved by using a 'look-ahead' rule for scheduling rather than FCFS. However, contrary to our original expectations, the extra sophistication of a GA-based scheduler was not in this case worth using. The SPT(all) rule also uses a 'look-ahead' principle, and in this case produced superior results to the GA, and produced them much faster. In any particular problem the actual time needed for scheduling must also be considered, so whether there is any gain over FCFS in practice will of course be problem-dependent.

The problem investigated is rather straightforward, so in a sense it is not surprizing that a simple rule like SPT(all) performs well. We intend to investigate more complex problems where more sophisticated approaches such as GAs might be needed.

We should also emphasize that in real problems processing times are not always known in advance, although we may be able to predict them with a fair degree of accuracy. Genetic algorithms have been found effective for stochastic flowshop sequencing [7], and future work will also investigate the potential for using GAs in multi-processor systems which are both dynamic and stochastic.

References

1. CLEVELAND, G. A. – SMITH, S. F. (1989): Using Genetic Algorithms to Schedule Flow Shop Releases. In J. D. Schaffer (Ed.) (1989) *Proceedings of the 3rd International Conference on Genetic Algorithms.* Morgan Kaufmann, Los Altos, CA.
2. CARTWRIGHT, H. M. – MOTT, G. F. (1991): Looking Around: Using Clues from the Data Space to Guide Genetic Algorithm Searches. In R. K. Belew and L. B. Booker (Eds.) (1991) *Proceedings of the 4th International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA.
3. REEVES, C. R. (1993): A Genetic Algorithm for Flowshop Sequencing. *Computers & Ops. Res.,* (to appear).
4. REEVES, C. R. – KARATZA, H. (1993): Dynamic Sequencing of a Multi-processor System: a Genetic Algorithm Approach. In R. F. Albrecht, C. R. Reeves and N. C. Steele (Eds.) (1993) *Proceedings of International Conference on Artificial Neural Networks and Genetic Algorithms,* Springer-Verlag, Vienna.
5. CONWAY, R. W. – MAXWELL, W. L. – MILLER, L. W. (1967): Theory of Scheduling. Addison-Wesley, Reading, Mass.
6. SAUER, C. H. – CHANDY, K. M. (1981): Computer Systems Performance Modelling. Prentice-Hall, New Jersey.
7. REEVES, C. R. (1992): A Genetic Algorithm Approach to Stochastic Flowshop Sequencing. *Proc. IEE Colloquium on Genetic Algorithms for Control and Systems Engineering.* Digest No. 1992/106, IEE, London.

PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 41, NO. 3, PP. 239-249 (1997)

A LOW-COST ROBOT CONTROLLER AND ITS SOFTWARE PROBLEMS

Gábor TEVESZ, István BÉZI and István OLÁH[1]

Department of Automation
Technical University of Budapest
H-1521 Budapest, Hungary
Phone: 36 1 463-2870, Fax: 36 1 463-2871
E-mail: tevesz@aut.bme.hu, bezi@aut.bme.hu, olah_i@aut.bme.hu

Received: July 4, 1997

Abstract

In recent years the need for advanced robot control algorithms for industrial robots has grown. The development of a low-cost robot controller to support the development, implementation and testing of those algorithms which require high computational power was targeted. This paper deals with the requirements of an experimental controller that can be connected to a NOKIA-PUMA 560 robot arm. It explains the IBM PC compatible host and the TEXAS Digital Signal Processor (DSP) based hardware. On the host computer the UNIX-like QNX real-time operating system is used. In the current phase of development the robot controller works with the classical decentralised joint control based strategy. The Advanced Robot Programming System (ARPS) explicit robot programming language is implemented.

Keywords: robot control, multiprocessor systems, IBM PC, DSP, QNX, ARPS.

1. Introduction

Numerous advanced robot control algorithms, such as computed torque technics (nonlinear decoupling and decentralised PID controllers), resolved motion acceleration control, hybrid position and force control (operational space formulation), model reference adaptive control and fuzzy control, are currently undergoing research and experiment. Such algorithms are increasingly in demand in the workplace. The hardware and software of industrial robots are closed systems, with an unsatisfactory computational power and thus the implementation of advanced control strategies is not feasible.

[1]The research work was supported by the Hungarian research Fund (OTKA) under the terms of grant No. T 016855

2. The Requirements of the Robot Controller

Advanced robot control algorithms are based on the nonlinear dynamic model of the robot arm with a driving torque of:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + (\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}(\mathbf{q}) = \tau, \tag{1}$$

where $\mathbf{H}$ is the inertia matrix, $\mathbf{C}$ is the centripetal, Coriolis and friction effects, $\mathbf{D}$ is the gravitational part and τ is the driving torque. An over-riding problem in the implementation of such algorithms is that the driving torque needs to be computed within 1–10 ms. This computation requires multiprocessor architecture and nonrecursive algorithms are favoured where parallel computing is possible (LANTOS, 1991 and SOMLÓ et al., 1997).

The low-cost experimental robot controller was initially developed for a six degrees of freedom NOKIA-PUMA 560 robot arm. The requirements of the controller hardware were as follows:

- modular architecture,
- extendibility,
- easy system development possibility,
- interfacing several sensor processors possibility,
- changing the host computer possibility.

The robot controller should:

- be an open system,
- have a modular (layered) structure that does not reduce efficiency,
- be based on such programming languages and methods that they guarantee software portability.

3. The Hardware

Examining the possibilities of the development of such systems the IBM PC based multiprocessor architecture was chosen (BÉZI and TEVESZ, 1994). Two robot controllers were built for the cooperating departments. One works at the Department of Automation while the other is used at the Department of Process Control.

Regarding the above requirements, the aim was to develop an open system where the implementation and testing of different control algorithms is easy. Capitalising on the benefits of commercial products, the available parts of the control system were chosen for their speed, serviceability and interchangeability. The whole control system was built on the IBM PC basis because of the many standardised, easily obtainable modules and

because of the wide-spread usage of these machines in the field of process control. The multiprocessor core of the system and the interface to the controlled process was target specific design. The result of this development is shown in *Fig. 1.*

The present host computer is based on an i486DX2/66 main board. The advantage of this choice is the extensibility of the performance by simply upgrading the host processor without major changes in the software system being developed. The link between the modules of the system is the ISA bus. This is the only bus system in the 12 year history of IBM PC that has survived all other improvements in more modern announced bus systems (EISA, VESA, VLB, PCI). Because of the number of used modules the host computer consists of two racks connected with ISA bus expansion cards without reducing the speed of the communication.

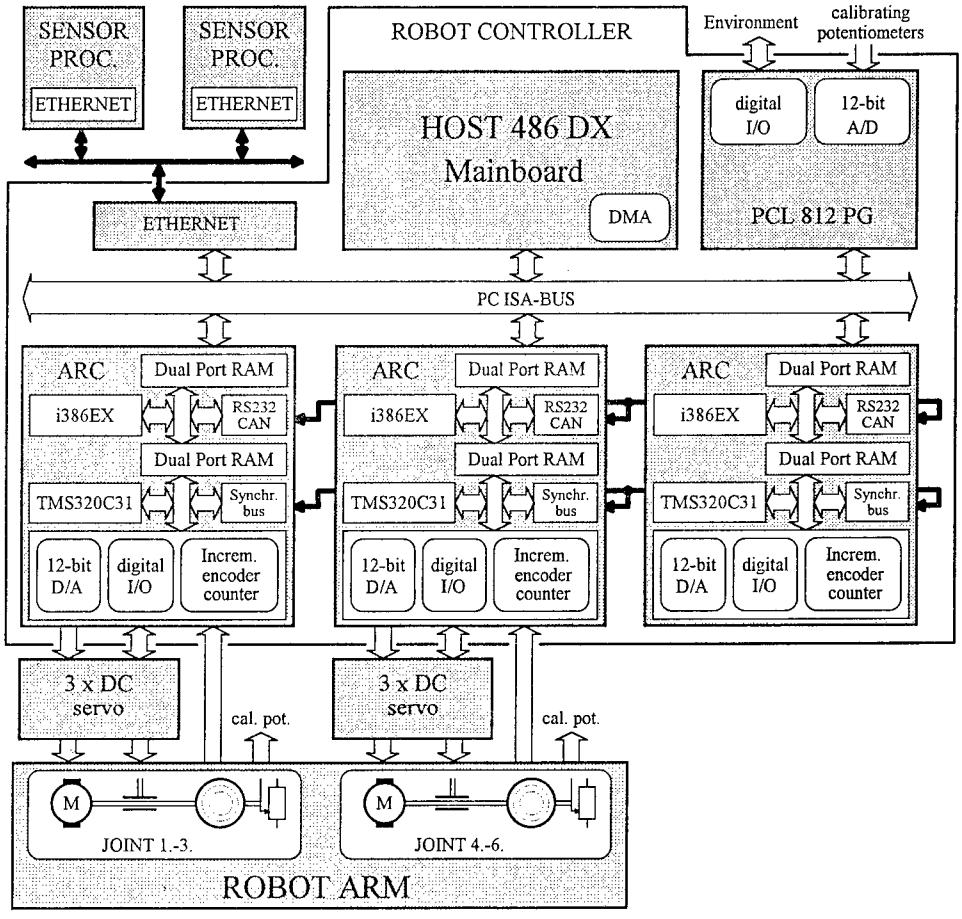


Fig. 1. Configuration of the robot controller

The connection between the host and the other intelligent units is provided by a Local Area Network. At present this is built up with a 10 Mbit/sec Ethernet line but there are no difficulties in changing it to the compatible type of 100 Mbit/sec speed on demand. The benefits of this choice are: standardised modules, developed continuously, newer and newer products are compatible with previous ones and it provides the spatial distribution of the units.

The connection between the host and the passive environment is provided by a high speed, high performance, multi-function data acquisition card (PCL-812PG). The calibrating potentiometers of the robot arm are connected to the programmable gain analog inputs. In addition the data acquisition card supports connecting of 16 digital input and 16 digital output lines that provides more control tasks to be made.

The algorithmic part of the robot control and the direct control of the arm is fulfilled by the self developed Advanced Robot Controller (ARC) card. This is the most important part of the system so the features are explained in detail. This card is a multiprocessor card which communicates with the host over the ISA bus, with each other over the CAN bus, a high speed serial bus and over a special parallel (synchronising) bus. This card provides all the signals for the external DC servo amplifiers and receives the sensor signals of incremental encoders in the robot arm.

To increase the computational performance a high-speed transputer card can be implemented in the system. The task of this card depends on the particular control algorithm. The aim is to communicate with this card over the ISA bus providing interchangeability with another, more suitable, task specific card regarding the unified communicational system. In the present system a third ARC card is used for the computational needs of the advanced control algorithm. The first two ARC cards are used for the direct control of axes.

The six joints of the robot arm are directed by the self developed DC servo amplifiers. These amplifiers work over 18 kHz switching frequency with pulse width modulation. The set point signals for current come from the ARC cards in analog form and the analog PI controllers of the servo amplifiers perform the real values (and limits on demand).

The most important part of the experimental robot control system is the ARC card. The block scheme of this module is shown in *Fig. 2.* Every card consists of two microprocessors: the so called preprocessing unit is an i386EX microcontroller while the second, so called joint processor, is a TMS320C31 DSP. On each card 4 whole featured interface is available for electronic drives (joints or axes). Considering the complex task of robot control and depending on the implemented algorithms the use of two or

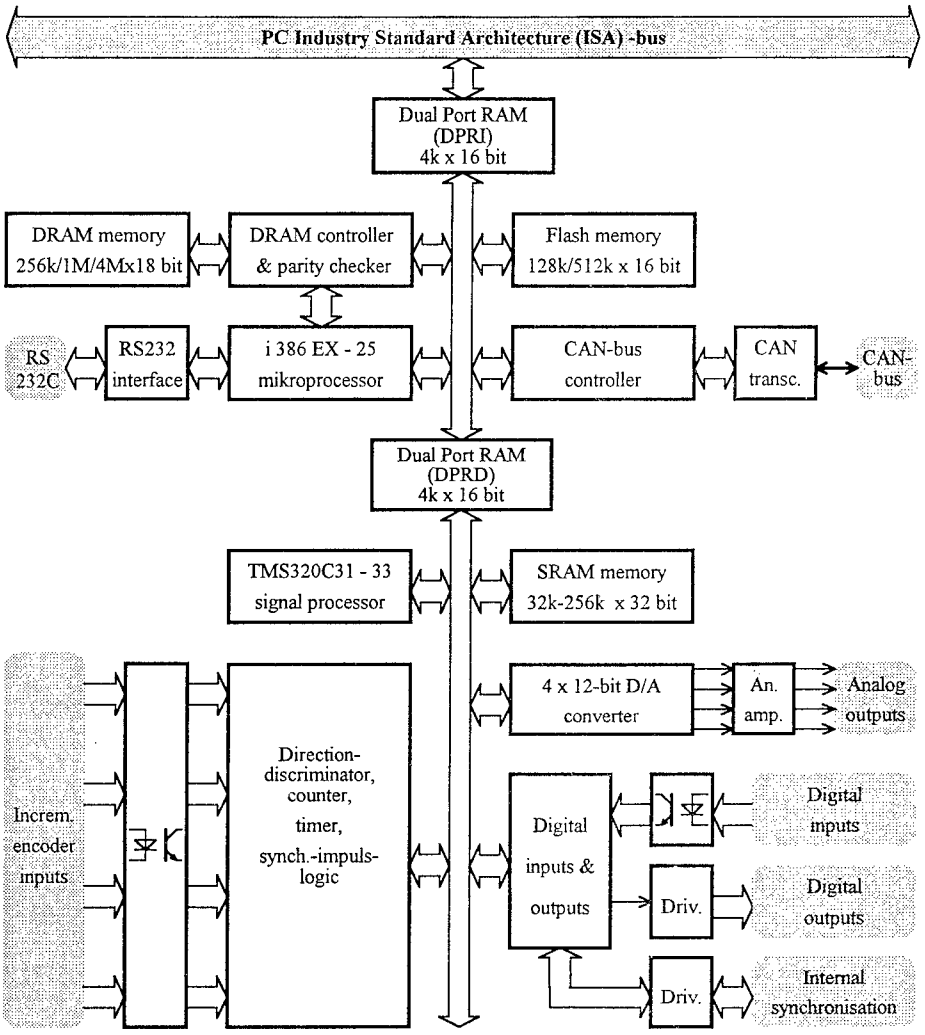


Fig. 2. Block scheme of the ARC board

three ARC cards is planned. This way each card provides the connection and control task for three or two joints.

The main blocks of the cards are as follows:

The preprocessing unit is the new embedded microprocessor by Intel (i386EX). This device provides a 100% compatible environment with PCs integrating the basic peripheral functions of a main board. This 32 bit, 80386 based microprocessor serves not only with the MMU for the possibility of Virtual Protected Mode but contains special enhancements like the

missed watch-dog circuit. The memory of this unit consists of a flash block (256 kbyte or 1 Mbyte) and the usual DRAMs (the capacity depends on the used SIM modules: from 512 kbyte up to 8 Mbyte). The non-volatile chip contains at present a BIOS but the implementation of a whole ROM-DOS system is planned which would make the development and testing of different robot control algorithms much easier.

The joint processing units are high-speed CMOS 32-bit floating-point single-chip Digital Signal Processors (DSP) - TMS320C31. These DSPs have the highest performance in the system with a capability of 16 MIPS and 32 MFLOPS. The tasks of these joint processors are:

- taking the position signals and calculating the speed and acceleration,
- credibility check using the null impulses,
- supervising the position, speed and acceleration limits,
- producing the current set point values for servo amplifiers,
- providing the synchronization in starting and stopping the axes.

The high performance of the joint processor is supported by the high-speed static RAM (32–128 kword). The required access time (18–20 nsec) is unreachable in the case of non-volatile memory so this device uses the built in Boot Loader to load its own program system from the flash of the preprocessing unit over the (dual port) DPRAM. The flash memory of the preprocessor contains the programs for both processors on the card. In order to the highest possible response time the signals of the position, speed and acceleration sensor incremental encoder are connected directly to the DSPs through the intelligent interfaces (direction discriminator, counter and null impulse logic) realised by tacho processors of type TC3005H. The analog circuits for current set point and the servo amplifier controlling digital input and output signals are also connected to the joint processor directly. A three-wire synchronisation channel provides for simultaneous movement of the axes connected to different ARC cards.

Communication Channels inside the System

The most important questions in time of the design were the choice of proper communicational channels. The low throughput of these connections would limit the performance of the responsible units.

These channels are shown in *Fig. 3*. The broken connections are for the most important development possibilities that are not implemented in the present system.

The preprocessing units are functionally set between two other processors (the host and the joint processor) and the communication in both

directions has a main role in the functionality. Since both of the connections have a high capacity demand the highest throughput has been chosen here: the links are high-speed dual port RAMs (DPRI and DPRD blocks in *Fig. 2*). This realisation provides the maximal speed communication because the communication is delayed only in the case when both sides access the same cell simultaneously. The throughput of the channels between the host and preprocessor using the built in 16 bit-word access is approximately 5 Mbyte/sec and between the preprocessor and joint processor is 10 Mbyte/sec. In order to attain optimal performance the available interrupt possibilities should be used on every side. 8 built in hardware semaphores per page are available for the data block consistency.

The preprocessors in communication with each other should not use the host (over the other way loaded ISA bus) but these modules are connected via a real multimaster serial Controller Area Network (CAN) bus. This bus is accessible on the back plane of the ARC cards so other external elements of the control system can be connected via CAN bus on demand. The physical speed of 1 Mbit, concerning the message identification and check philosophy of CAN protocol, provides approximately 50 kbyte/sec data throughput. Of course this channel is not for moving huge data blocks but rather for changing intermediate results of robot control algorithms. The multimaster property of CAN bus providing the communication of equal members is essential (however, the traffic can be mastered by one).

The RS232C compatible serial line of the preprocessor has been built only for test purposes in the early development phase.

A special synchronisation bus has been developed for the joint processors providing the simultaneous movement of joints. This bus consist of only three lines with the availability of wired OR (in negative: AND) logic between the processors. Its speed is ca. 8 Mbit/sec.

5. Software

The system was designed to be low-cost so the heterogeneous architecture caused some informatical problems. The requirements were:

- software development and test on host,
- connection with intelligent sensors via network,
- user friendly operator interface,
- software development and test for the preprocessor (i386EX),
- program transfer and running on the preprocessor,
- software development and test for the joint processor (TMS320C31),
- program transfer and running on the joint processor.

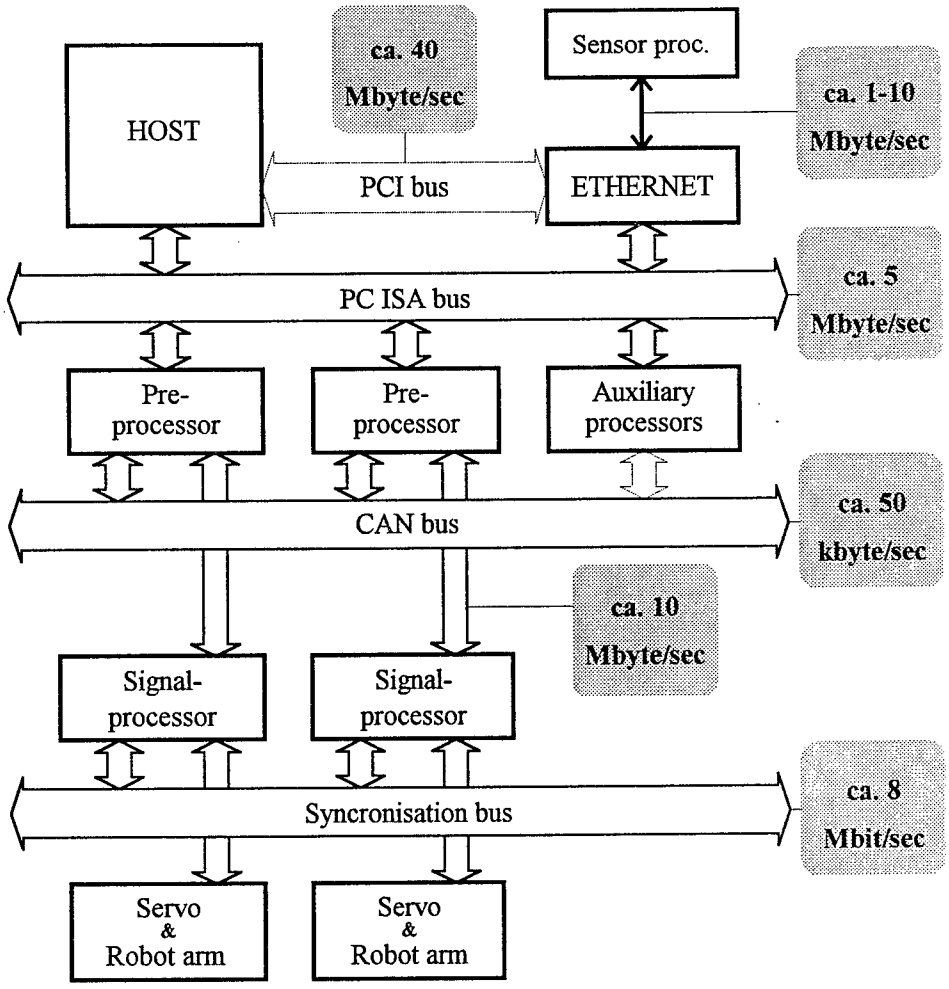


Fig. 3. Communication channels inside the system

In the host computer the QNX real-time UNIX-like operating system was chosen (DODGE, 1992). This provides multitasking, priority-driven preemptive scheduling with three scheduling algorithms following the IEEE POSIX 1003.4 specification. The QNX is a flexible, distributed (networked) operating system, characterised by microkernel architecture and message-based interprocess communication. This interprocess communication is performed in three ways: as messages (synchronous communication), proxies (special form of messages) and signals (asynchronous communication).

Another communication possibility is shared memory when proxies and signals can provide the synchronisation. The QNX has advanced timing facilities and the interrupts can be handled on a process level. The QNX Windows system provides the user friendly interface.

In the interests of portability and transferability of particular tasks between processors the software development uses only C language (Watcom/Borland/ Microsoft/Texas Instruments). The benefits of this 'broad-spectrum' (high and low level capabilities) language are well-known and C is available for all used microprocessors.

The first phase of software development for ARC card provides a simple BIOS. With the help of this system the program transfer and running is possible through few system calls (software interrupts on i386EX and assembly function calls on TMS DSP). In addition to the previous functions the possibility of character input and output is provided for both processors. Based on the BIOS a developing environment was made which is used for program and test each processor using C language. The preprocessor is able to run programs in DOS EXE format but does not provide the whole IBM PC environment. The startup code of Borland C was changed and the same type of modification was made in case of the Watcom C compiler. In order to achieve proper functionality of the watch-dog circuit the boot routine (c_int00) of the Texas Instruments C compiler for TMS320C3x/C4x processors had to be modified as well. This compiler is available only for DOS so the QNX Rundos (a DOS emulator) was used first. Because of the incompatibility between Rundos and TMS C compiler this method was cancelled (VÖRÖS, 1995).

For the build up and first time check of the whole hardware system the following programs were written for QNX Windows:

- set and measurement of the analog and digital ports on the PCL-812PG card. This program is able for displaying the signals of the potentiometers in the robot arms,
- program transfer and running in the joint processor using the preprocessor only for data forwarding,
- determination of joint position and speed giving out the current set point and sending back the information (including ARC status) to the host.

Based on the experience of test programs low level control software was made which realizes a double loop digital PID controller for speed and position (the third loop is the analog PI controller in the DC servo amplifiers). The communication protocol of the undocumented teach pendant was decoded and the driver for this device was implemented in the software. Other software modules are available for determining the character-

istic of potentiometers and calibrating the robot arm (DONÁT, 1996 and KORCSMÁR, 1996.).

The Advanced Robot Programming System (ARPS, the original programming system of the NOKIA-PUMA 560) was implemented in the Department of Process Control (JÁSZ, 1996) and integrated into the software system (KORCSMÁR, 1996).

6. Conclusions

The experimental robot control system in its present form is serving well in the education of robot programming. Considering the present experience for the realization of advanced robot control algorithms the next steps are:

- developing the BIOS system of the ARC card burning some new functions into flash memory,
- speeding up the communication over dual port RAMs,
- working out the communication on CAN bus,
- implementing an operating system subset (utilizing the embeddable capabilities of QNX microkernel architecture) on the preprocessor.

The present experimental robot control system on the Department of Process Control will be used as part of an intelligent control system of a robot with a dextrous hand. The development of the software based on the above architecture is being continued at the Department of Process Control in some new directions like a new communication principle between host PC and ARC control boards, new basic software components, advanced control algorithms using self tuning adaptive control, neural control and fuzzy experts (KLATSMÁNYI, 1996). The architecture and the software of two further subsystems were developed at the Department of Process Control (LANTOS et al., 1997). The first of these new subsystems is the control system of the tendon-driven three-fingered (9 degree-of-freedom) TUB dextrous hand (LUDVIG, 1996a), allowing high-level grasp-planning and implementation (LUDVIG, 1996b). The second new subsystem is a low-cost stereo vision system. The task of the stereo vision system is to collect image information about the robot, the hand and the environment, process it and send the results to the control systems of the robot and the hand. The image processing is based on the theory of projective geometry, statistical object recognition and parameter identification (LANTOS et al., 1997).

References

1.BÉZI, I. – TEVESZ, G. (1994): PC Based Robot Controller for Advanced Control Algorithms. *Proc. of the Joint Hungarian - British International Mechatronics Conference,* Budapest, September 21-23, 1994, pp. 743-748.

2.BÉZI, I. – TEVESZ, G. (1996): Komplett aus Standard-Komponenten. *Elektronik*, 1/1996, pp. 44-48.

3.DODGE, D. (1992): The Continuing Evolution of QNX. *QNX News,* No. 6, pp. 7-15.

4.DONÁT, T. (1996): Kísérleti robotvezérlő élesztése és tesztelése (Setting up and Testing of an Experimental Robot Controller). M. Sc. Thesis at the Department of Automation, Technical University of Budapest (in Hungarian).

5. JÁSZ, G. (1996): Robotprogramozási nyelv kifejlesztése QNX valósidejű operációs rendszerhez (Development of a robot programming language under QNX real-time operating system). M.Sc. Thesis at the Department of Process Control, Technical University of Budapest (in Hungarian).

6. KLATSMÁNYI, P. (1996): Design and Implementation of a New Communication Concept and its Application in the New Version of the Control System of a PUMA Robot. OTKA T 16855/3 Research Report at the Department of Process Control, Technical University of Budapest.

7.KORCSMÁR, A. (1996): Teszt és működtető programok kifejlesztése célhardverekhez (Development of Testing and Operating Programs for Special Hardware). M. Sc. Thesis at the Department of Automation, Technical University of Budapest (in Hungarian).

8. LANTOS, B. (1991): Robotok irányítása (Robot control). Akadémiai Kiadó, Budapest (in Hungarian, 2nd edition 1997).

9. LANTOS, B. – KLATSMÁNYI, P. – LUDVIG, L. – TÉL, F. (1997): Intelligent Control System of a Robot with Dextrous Hand. *Proceedings of the IEEE International Conference on Intelligent Engineering Systems INES'97*, Budapest (being appeared).

10. LUDVIG, L. (1996a): Intelligent Control of Dextrous Robot Hands. *Proceedings of the 5th International Workshop on Robotics in Alpe - Adria - Danube Region RAAD'96*, Budapest, pp. 345–350.

11. LUDVIG, L. (1996b): A Practical Approach to the Choice of Optimum Grasp for Three-fingered Dextrous Hands. *Proceedings of the 27th International Symposium on Industrial Robots ISIR'96*, Milan, pp.459–464.

12. SOMLÓ, J. – LANTOS, B. – CAT, P. T. (1997): Advanced Robot Control. Hungarian Academic Press, Budapest (in press).

13. VÖRÖS, I. (1995): Programfejlesztői környezet kialakítása kísérleti robot vezérlőhöz (Design of Program Developing Environment for an Experimental Robot Controller). M. Sc. Thesis at the Department of Automation, Technical University of Budapest (in Hungarian).

INFORMATION FOR AUTHORS

Submitting a Manuscript for Publication. Submission of a paper to this journal implies that it represents original work previously not published elsewhere, and that it is not being considered elsewhere for publication. If accepted for publication, the copyright is passed to the publisher: the paper must not be published elsewhere in the same form, in any language, without written consent of the executive editor. The first author will receive 50 free reprints.

Manuscripts should be submitted in English in two copies to the editors' office (see inner front cover). Good office duplicated copies of the text are acceptable.

Periodica Polytechnica is typeset using the TEX program with the AMSTEX macro package. Therefore, authors are encouraged to submit their contribution after acceptance in this form too, or at least the text of the article in a simple ASCII file (e. g. via email or on a floppy diskette, readable on an IBM compatible PC). By this solution most of the typesetting errors can be avoided, and publishing time can be reduced. An AMSTEX style file with a sample TEX source file is also available upon request.

Compilation and Typing of Manuscripts. Contributions should be typed or printed in double spacing (24 pt spacing when using text processors), on A4 paper. One page may contain not more than 10 corrections (paints do not count).

The maximum length of the manuscript is 30 standard pages (25 lines, with 50 characters in a line), including illustrations and tables. When more characters are typed on a page, the allowed page number is reduced accordingly.

When using a text processor, please use a (preferably English) spelling checker before the final printing.

Use one side of the sheet only. Paragraphs are to be indented by 5 spaces and not to be preceded by a blank line.

A correctly compiled manuscript should contain the following items:

(1) Title page giving: the suggested running header (max. 50 characters) for the odd typeset pages; the title (short, with subheading if necessary); the name(s) of the author(s); affiliation (institution, firm etc.) of the author(s), in English, with mailing address, telefax and phone numbers and email address; grants, scholarships etc. during the research (in a footnote); an informative abstract of not more than 200 words with 3-5 keywords;
(2) Textual part of the contribution with a list of references;
(3) A separate sheet listing all figure captions and table headers;
(4) Illustrations and tables (please put your name on each sheet), at least one set of illustrations in very good quality for reproduction.

Abstract. A summary or abstract of about 100-200 words should be provided on the title page. This should be readable without reference to the article or to the list of references, and should indicate the scope of the contribution, including the main conclusions and essential original content.

Keywords are to be given for the purpose of data bases and abstracting; avoid too general keywords which provide no help in literature searching.

General rules for the text. Chapters are to be numbered with Arabic numbers in decimal hierarchy.

Wherever possible, mathematical equations should be typewritten, with subscripts and superscripts clearly shown. Metric (SI) units are to be used, other units may be given in parentheses. Equations must be numbered on the right side, in parentheses. Handwritten or rare mathematical, Greek and other symbols should be identified or even explained if necessary in the margin. Letters denoting quantities are to be distinguished by different setting both in the formulae and in the text. Remember the rule that scalar quantities are to be denoted by italics (underline by hand in your manuscript), vectors by lower case bold type letters (underline doubled), and matrices by bold capitals (underline doubled). Dimensions (like cm, Ohm, V etc.) and standard function names (sin, ln, P etc.) are to be typeset in Roman typefaces (not in italics). A few important words may be distinguished by italic setting (underline).

Illustrations and Tables. Graphs, charts and other line illustrations should be drawn neatly in Indian ink, or printed by a laser printer. Computer printouts can only be used if they are of excellent quality. Figures should be submitted in an adequate size for camera-ready pages in size 1.2:1. Suggested line thicknesses: 0.18–0.35–0.4 mm or 0.5–0.7–1.14 pt. Letter sizes: 0.4 mm (10 pt). All figures should be numbered with consecutive Arabic numbers, have descriptive captions, and be referred to in the text. Captions should be self-explanatory, not merely labels. Figures must not contain lengthy texts; use captions instead.

Number tables consecutively with Arabic numbers and give each a descriptive caption at the top. If possible avoid vertical rules in tables. Tables should be preferably submitted in camera-ready form.

The earliest acceptable position of figures and tables is to be indicated in the left margin.

References. In the text references should be made by the author's surname and year of publication in parentheses, e.g. (Earl et al, 1988) or ...was given by Kiss and Small (1986a). Where more than one publication by an author in one year is referred to, the year should be followed by a suffix letter (1986a, 1986b etc.), the same suffix being given in the reference list. For the style of the reference list, which is to be given in alphabetical order, see the examples below for journal articles, conference papers and books.

Earl, J., Kis, I. and Török, I. (1988): Partial Discharge Measurement in Cables. *Periodica Polytechnica Ser. Electrical Engineering*, Vol. 32, No. 4, pp. 133–138.

Kiss, S. and Small, A. B. (1986a): Roundoff Errors in FFT. *Proc. 5th IEEE Symposium on Signal Processing*, Boston (MA), May 3–5, 1986. New York, NY, IEEE Press, CH0092-2875/86, pp. 3.5–3.9.

Kiss, S. and Small, A. B. (1986b): Ellenállások (Resistances). Budapest, Tankönyvkiadó. pp. 533–535. (in Hungarian)

More detailed guidelines for authors, with hints for the preparation of figures and with a sample page, are available from the editors' office (see inner front cover).

CONTENTS